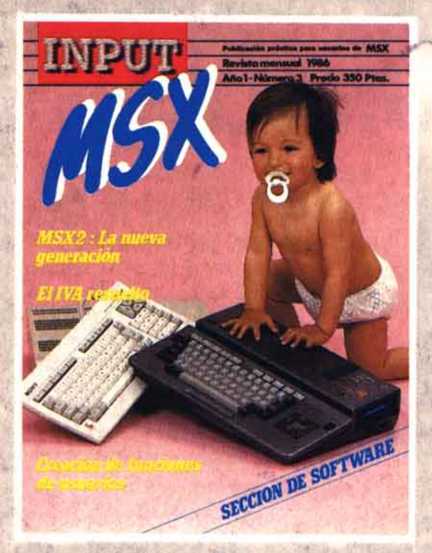


INPUT
Publicación práctica para usuarios de MSX
Revista mensual 1986
Año 1 - Número 3 Precio 350 Ptas.
MSX
MSX2 : La nueva generación
El IVA resuelto
Creación de funciones de usuarios
SECCION DE SOFTWARE
PHILIPS



AÑO 1 NUMERO 3

DIRECTOR:
Alejandro Diges
DIRECTOR TECNICO:
Roberto Menéndez
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, Benito Román, Esther de la Cal, Ernesto del Valle, Equipo Molisoft.

INPUT MSX es una publicación juvenil de EDICIONES FORUM

GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ Gral. Varela, 35, 3.º-11
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Sirven Grafic
C/ Gran Vía, 754-756. 08013 Barcelona
Depósito legal: M. 27.884-1985

SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Alberto Alcocer, 46, 4.º
28016 Madrid. Teléf. 250 10 00

DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

Se ha solicitado el control OJD

INPUT MSX es independiente y no está vinculada a MSX Research o sus distribuidores.

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas.

INPUT MSX

SUMARIO

LOS LIOS GENERACIONALES

En los últimos tiempos se viene hablando mucho de los ordenadores de la nueva generación. Pero quizás son los **MSX2** los que más pueden impactar en el usuario no profesional. De hecho acaban de hacer su aparición en nuestro mercado y ya se les vislumbra un futuro prometedor.

Al hablar de esta nueva generación estamos pensando también en máquinas como el **Amiga,** de **Commodore,** y el **520 ST,** de **Atari,** siguiendo la escuela que marcase en su día el sistema **Star,** de **Xerox,** con su nueva aproximación al manejo más intuitivo y simplificado por parte del usuario final, entiéndase varias ventanas de vídeo en una misma pantalla, el uso generalizado del «ratón», mayores capacidades gráficas, etc. Después llegó para asombro de todos el sistema **Lisa,** de **Apple,** caro predecesor del popular **Mackintosh.**

Los sistemas que responden al estándar **MSX2** siguen utilizando un microprocesador de 8 bits, frente al potente **68000** de los otros sistemas aludidos, pero su especial fortalecimiento en cuanto a capacidades gráficas no les producen envidias.

La mayor disponibilidad de memoria que en muchos otros sistemas de 8 bits y la superior velocidad de proceso, le hacen ocupar de momento al **MSX2** un segmento de mercado a medio camino entre los ordenadores domésticos de bajo precio (**MSX** primera generación, **Spectrum, Amstrad** o **Commodore 64**) y los ordenadores de gestión del tipo **PC.**

El ajuste final en su precio de venta —que actualmente oscila entre unas ciento cincuenta y unas trescientas mil pesetas— junto a la cantidad y calidad del *software* que irá apareciendo, serán los factores determinantes para el éxito en la previsible carrera por situarse a la cabeza de un segmento muy reñido en el próximo futuro. Estamos viviendo tiempos de transición y, si bien existe todavía un importante hueco de mercado en nuestro país, es probable que asistamos a una reorganización en los precios de las máquinas de la primera generación, aumentando su popularidad entre los potenciales usuarios que están por decidir cuál será su primer ordenador.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Alberto Alcocer, 46 - 4.º B. 28016 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido ____________________

Segundo título elegido ____________________

Tercer título elegido ____________________

Programa que te gustaría conseguir ____________________

Qué ordenador tienes ____________________

Nombre ____________________

1.er Apellido ____________________

2.º Apellido ____________________

Fecha de nacimiento ____________________

Teléfono ____________________

Dirección ____________________

Localidad ____________________

Provincia ____________________

INPUT MSX N.º 3



DALE LA MANO AL ROBOT

Que te parece la idea de disponer de un brazo robot que, ahora que aprieta el calor, te acerque un vaso con tu refresco preferido, o estreche la mano de tus amigos al ser presentado? Spectravideo te ofrece estas y muchas más posibilidades con su brazo robot Quickshot SVI 2000. Este robot, en plastico amarillo, tiene una base que soporta el conjunto brazo-antebrazo-mano. Esta última puede ser una pinza para coger objetos o un iman. Hay dos posibilidades para trabajar con el; en la primera, el control se lleva a cabo utilizando dos *joysticks*, conectados a dos conectores tipo D de la base del robot. De esta forma se puede determinar el giro de cualquiera de los cinco ejes que determinan el movimiento (5 grados de libertad). En este caso no hay conexión al ordenador, y lo único que hace falta son las pilas de alimentación del brazo.
La otra posibilidad es utilizar el cartucho ROM que permite la conexión al *port* de cartuchos de cualquier MSX. Este tiene en su parte superior dos conectores para *joystick* que se conectan a los que hay en la base del brazo. Una vez hecha la conexión se puede hacer uso de Rogo, un lenguaje de programación, similar en algunos aspectos al Logo, que incorpora 48 comandos para control de movimientos.
Spectravideo nos ha anunciado que comercializará el brazo probablemente a mediados de Julio. El precio esta todavia por determinar.



COMPULOGICAL, NUEVOS REPRESENTADOS

La firma distribuidora de *software* Compulogical ha firmado nuevos contratos de representación exclusiva con el fabricante britanco Kuma, que dispone de interesantes títulos para MSX. En primer lugar lanzará cuatro títulos. Los títulos de Acosoft: Jetbomber, Oh Shif, Hooper y Boom, serán los primeros en alcanzar nuestras tiendas como producto de la firma con esta casa.



UN CARTUCHO QUE TE AYUDA

Grabados en una EPROM, en el interior de un cartucho, están los datos de los diferentes comandos BASIC. Se trata de un proyecto desarrollado por el Servicio Central de Mitsubishi, en Barcelona.
El cartucho, que ha sido bautizado con el nombre de Tutor, esta pensado para los programadores inexpertos. Cuando surge alguna duda, el programador no tiene más que hacer una llamada al cartucho, mediante CALL. El cartucho entra en acción y nos pregunta por el comando que buscamos. Al responderle, aparece en pantalla toda la información que sobre dicho comando hay en la EPROM (sintaxis del comando, función, etc). Al terminar la consulta, el cartucho devuelve control al BASIC y queda a la espera. El cartucho hace las veces de manual de consulta con la ventaja de que el acceso a la información es más rápido y no requiere cambiar de actividad. Ademas, el programador no pierde ni un solo *byte* de memoria, ya que el cartucho no interfiere con la RAM.



MUY PRONTO CD-ROM

Sony, uno de los líderes, junto con Philips, en la tecnología CD-ROM planéa la integración de un sistema de lectura-escritura CD-ROM para las próximas versiones MSX. Se baraja como probable la fecha de las navidades de 1987 para la aparición de los primeros sistemas comerciales basados en esta nueva tecnología.
Los equipos que incorporen el sistema CD-ROM, en el que la capacidad de almacenamiento se mide por centenares de Megabytes (los diskettes almacenan centenares de Kilobytes), irán destinados, en principio, al mercado profesional y de gestión. No obstante, firmas de software como Aackosoft, en Holanda, tienen puestas sus miras en la utilización de la tecnología CD-ROM también en el área de los juegos. La idea es la del almacenamiento masivo de imágenes digitalizadas y su posterior presentación en pantalla a la suficiente velocidad.



SUSTRAIDO UN MSX 2

El pasado mes fue sustraido del interior de un automovil un ordenador Sony MSX 2, HB-F500P. Por ser un modelo aún poco difundido es facilmente identificable.
Se gratificará a quien de una pista que lleve a su recuperación. El número de serie es:400286
Llamar a la redacción de INPUT.

EL BUZON DE INPUT



En vuestro número 1 leí un artículo que hablaba de la constitución de la sociedad SVI España. **Yo tengo un SVI 328 y estoy interesado en el SV 606, adaptador** MSX **para mi** Spectravideo. **Este adaptador hasta ahora no se comercializa en España. Mi pregunta es: ¿Comercializará esta sociedad este adaptador? y, si no es así ¿de qué forma podría adquirirlo? Como este adaptador convierte al** Spectravideo **en un** MSX **compatible al 100%, estoy seguro de que muchísimos usuarios de los SVI 328 y 318 agradecerían esta solución.**

También tengo entendido que existe un cartucho de 80 columnas (creo que es el SV 727).

Francisco Fernández
Parque Mediterráneo, 11
Málaga

Spectravideo España *nos ha comentado que dispone de algunas unidades del adaptador SV 606, por lo que te aconsejamos que te pongas en contacto con dicha empresa para reservar el tuyo.*

Por lo que sabemos, este cartucho está pensado fundamentalmente para poder utilizar los juegos ***MSX****, sobre todo los de cartucho ROM, ya que en el caso de utilizar* cassette *te vas a encontrar con sólo 12K de memoria libre. Esto significa que no llegarás al cien por cien de compatibilidad que nos comentas.*

En cuanto al cartucho de 80 columnas, existe uno para tu máquina, pero su referencia no es la que mencionas. La verdadera referencia es SV 806. Este cartucho, para cuya conexión vas a necesitar del Superexpander o bien del Miniexpander SV 602, te va a permitir trabajar directamente en 80 columnas.



Siempre me he preguntado si no existen *pokes* en el sistema MSX, **y ya no sé por donde buscar.**

Por ejemplo, en el programa Jet Set Willy 2 **te puedes volver loco para avanzar, porque no hay *pokes*, ni mapas. ¿Podrían dedicar algún apartado a este tema?**

David Botella Polo
Infanta Mercedes, 53
Madrid

El BASIC de los ordenadores ***MSX*** *incluye la instrucción POKE, que permite introducir un valor entre 0 y 255 en una posición de memoria RAM. Dicho con tus palabras: en el sistema* ***MSX*** *sí existen POKEs.*

Otra cosa es saber sobre qué posiciones o direcciones de memoria hay que actuar, con POKE o de cualquier otra forma, para conseguir vidas infinitas en este juego, o para evitar las colisiones entre sprites *en aquel otro, etc.*

Sabemos que hay mucho interés por estos temas relacionados con los videojuegos y, por ello, les dedicaremos la debida atención, bien a través de una sección específica o quizá dentro de la sección Revista de Software.

De todos modos, en este tema, como en cualquier otro, nos gustaría contar con la colaboración de nuestros lectores.

Así que ánimo, seguro que habrá premios para todos.



Me gustaría saber si en INPUT **vendéis los programas que salen anunciados en la revista y de ser así, pediros que me enviéis una lista con los precios y la dirección de donde tengo que pedirlos. A lo mejor podéis publicar la lista en algún número de la revista.**

Antonio Moreno
Madrid

Al responder a tu carta lo hacemos pensando no sólo en ella sino en muchas otras que hemos recibido de contenido similar.

En ***INPUT*** *no vendemos ningún tipo de material informático, ni periféricos, ni accesorios, ni programas. La publicidad que aparece en nuestras páginas proviene de los fabricantes e importadores, que con ella, pretenden daros a conocer sus productos. Si deseáis adquirir alguno de dichos productos, tenéis que dirigiros a los propios fabricantes, a través de los cupones que se incluyen en las páginas de publicidad o bien directamente. Ellos os darán toda la información que necesitéis.*

También podéis dirigiros a la tienda en la que habitualmente compréis material informático.



Hola amigos. Tengo un Philips MSX **y quisiera saber para qué sirve la tecla SELECT, ya que hasta ahora nunca la he usado.**

José Luis Martínez
Gijón

En las actuales versiones ***MSX****, la tecla SELECT no lleva asociada ninguna función. Sin embargo se puede utilizar desde cualquier programa, teniendo en cuenta que el código ASCII que le corresponde es 24 (&H18). Por ejemplo este listado...*

```
10 A$=INKEY$:IFA$=""THEN10
20 IFASC(A$)=24THEN PRINT
   "HOLA" ELSE GOTO 10
```

hace que el programa imprima HOLA, sólo cuando se pulsa la tecla SELECT.



737 FLIGHT SIMULATOR

EL MEJOR SIMULADOR DE VUELO CREADO PARA MSX.

DISEÑADO POR UN PILOTO PROFESIONAL DE UN 737.

¡UN AUTENTICO RETO!

Si están agotados en tu tienda habitual ¡¡LLAMANOS!!

Si deseas información y participar en los importantes sorteos que ZAFICHIP celebrará durante el año. . . ¡ESCRIBENOS!



ZAFIRO SOFTWARE DIVISION
Paseo de la Castellana, 141. 28046 Madrid
Tel. 459 30 04. Tel. Barna. 209 33 65. Télex: 22690 ZAFIR E



MSX2: LA NUEVA GENERACION

Claros exponentes de la evolución, no sólo del estandar MSX, sino de los ordenadores personales en general, ya están aquí los **MSX2**.

Hemos tenido ocasión de probar dos estupendos ejemplares de esta nueva generación de ordenadores; el **VG-8235** de **Philips** y el **HB-F500P** de **Sony**.



Una de las primeras críticas que recibió el estandar **MSX** fue la de que su *hardware* no era ni mucho menos revolucionario. O había una evolución dentro del estandar, o muy pronto los **MSX** de la primera generación, aún con todas las ventajas de la compatibilidad, iban a quedar obsoletos.

La evolución se produjo y nacieron los **MSX2**, con un *hardware* renovado que los sitúa a la cabeza en la actual tecnología de los ordenadores personales, en competencia directa con máquinas como el **Atari 520ST** o el **Commodore Amiga**.

Pero, y aquí está lo importante, la evolución se ha producido dentro del estandar, lo que significa que hay una total compatibilidad entre los modelos de distintas marcas, pero además se ha logrado mantener la compatibilidad con los **MSX** de la primera generación, de los que se puede aprovechar todo el *software* y *hardware* existente.

POR FUERA Y POR DENTRO

Las mayores diferencias entre estos dos modelos de la segunda generación están en su estética, en su aspecto exterior. **Philips** ha optado por un diseño compacto. El teclado, de 73 teclas, similar al de los modelos primera generación, y la unidad de *diskettes* de 3.5 pulgadas, de 360K en simple cara-doble densidad, van incorporados en la carcasa del ordenador. El teclado ofrece la particularidad de ser ajustable; actuando sobre un par de seguros es posible variar su inclinación. La parte posterior incluye un montón de conectores (*cassette*, impresora, monitor, TV, unidad de *diskettes* externa, conector audio/video con salidad RGB y segundo *slot* para cartuchos). En el lateral derecho están los conectores de los *joysticks* y la unidad de *diskettes*.

Por su parte **Sony** ha preferido el diseño modular. Por un lado el teclado, con un aspecto muy profesional (90 teclas y teclado numérico separado) y por otro la unidad central. En la parte frontal de esta última nos encontramos con la unidad de *diskettes* de 3.5 pulgadas, de 720K en doble cara-doble densidad, con los dos *slots* para cartuchos y con los conectores para *joysticks*. En la parte posterior están las conexiones para *cassette*, impresora, salida de audio/video, salida RGB, etc. Como no hay salida para TV, será necesario disponer de un monitor. En conjunto ambos modelos, cada uno en su estilo, resultan estéticamente muy agradables.

SOBRESALIENTE EN GRAFICOS

El nuevo *chip* de vídeo de los **MSX2**, que ha sido desarrollado para esta nueva generación por **ASCII Corporation, Microsoft** y **Yamaha**, lleva las siglas **V9938**.

Ofrece unas posibilidades impresionantes a través de 8 modos de pantalla diferentes, 512 colores y la posibilidad de acceder y manejar 128K de RAM de vídeo (**VRAM**) destinados exclusivamente a almacenar información gráfica.

En la figura 1 hemos incluido las características de cada uno de los modos de pantalla. En cuanto a los resultados que pueden obtenerse con el *chip*, basta con echar un vistazo a las

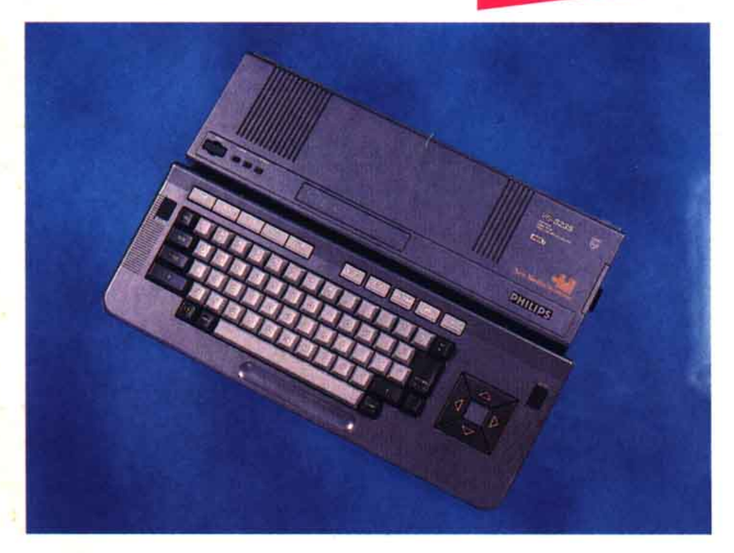
MODOS DE PANTALLA

No.	MODO	RESOLUCION	COLORES	PAGINAS	SPRITES
0	Texto	Maximo 24 lineas de 80 caracteres	16 simultaneos (paleta de 512)	--	No
1	Grafico	Maximo 24 lineas de 32 caracteres	16 simultaneos (paleta de 512)	--	Si (Simples)
2	Grafico	256x192 puntos	16 simultaneos (paleta de 512)	--	Si (Simples)
3	Grafico	64x48 multicolor	16 simultaneos (paleta de 512)	--	Si (Simples)
4	Grafico	256x192	16 Simultaneos (paleta de 512)	--	Si (Avanzados)
5	Grafico	256x212	16 Simultaneos (paleta de 512)	2 (VRAM 64K) 4 (VRAM 128K)	Si (Avanzados)
6	Grafico	512x212	4 Simultaneos (paleta de 512)	2 (VRAM 64K) 4 (VRAM 128K)	Si (Avanzados)
7	Grafico 128K VRAM	512x212	16 Simultaneos (paleta de 512)	2	Si (Avanzados)
8	Grafico 128K VRAM	256x212	256 Simultaneos	2	Si (Avanzados)

fotografías que acompañan a este artículo.

Como aspectos más destacables mencionaremos los siguientes: En primer lugar se puede trabajar en 80 columnas de una forma directa. En segundo lugar, se dispone de una resolución máxima de 512×212 puntos con 16 colores simultáneos en pantalla. Para los que quieran más colores, el modo 8 permite trabajar hasta con 256, simultáneos y a una resolución de 256×212 puntos. El manejo del color se lleva a cabo mediante la elección de las intensidades de cada uno de los tres colores primarios (Rojo, Verde y Azul), lo que ofrece una enorme flexibilidad, comparable a la de algunos sistemas de diseño gráfico profesionales. Además de la resolución, el *chip* **V9938** ofrece sobre todo velocidad de proceso, incorporando una serie de instrucciones para: transferir bloques de una zona a otra de la memoria de pantalla, manejar hasta 32 planos de *sprites* y utilizar hasta 4 páginas gráficas diferentes, a la máxima resolución, que pueden visualizarse de forma independiente y que permiten conseguir estupendos efectos de animación.

Hay que tener en cuenta que todo el manejo de gráficos se puede llevar a cabo desde programas BASIC, de





una forma sencilla y a mucha velocidad, gracias a los nuevos comandos gráficos incorporados en el BASIC **MSX** (Versión 2.0).

Un aspecto sobre el que se habían creado muchas expectativas, el del manejo de imágenes de vídeo (digitalización, superposición, etc.) no ha sido contemplado en ninguno de los dos modelos. Es seguro que se incluirá en modelos posteriores de esta segunda generación y que, para los modelos que estamos considerando, aparecerán cartuchos con el *hardware* y el *software* necesarios.

ADEMAS DE LOS GRAFICOS

Los ordenadores de esta segunda generación siguen utilizando la CPU **Z80A**, de 8 bits. Quizá sea este el aspecto que más incognitas planteó a la hora de la definición de estas nuevas máquinas. Desde luego la **Z80** está muy bien aprovechada pero, ¿Podrá competir con la potencia de las CPUs de 16 bits?

Por el momento está claro que sí y es probable que si las cosas llegan a ponerse feas para la **Z80A**, nos encontremos ya ante una tercera genración de **MSX**.

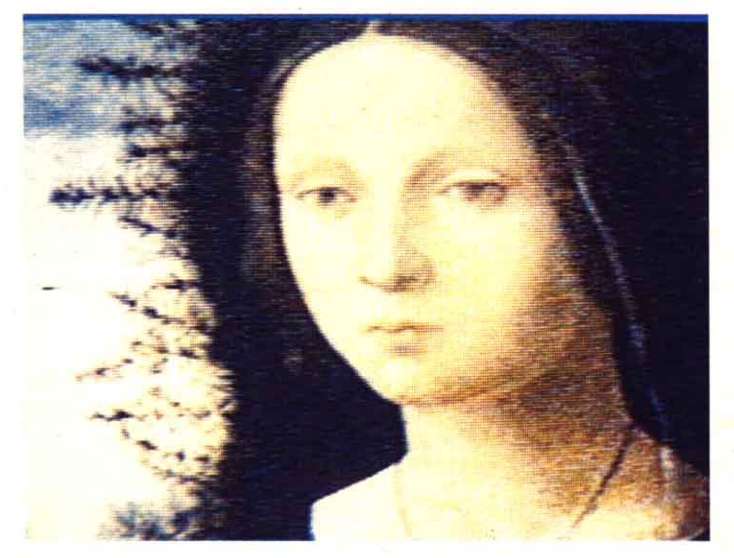
En el apartado memoria hay 128K de RAM en el caso de **Philips** y 64K en el de **Sony** (ambas máquinas tienen 128K de VRAM para uso exclusivo del *chip* de vídeo). En ambos casos hay 48K de ROM con la versión BASIC 2.0 y el sistema operativo.

El *chip* de sonido es de características muy similares a las del conocido **AY-3-8910** de la primera generación (8 octavas, 3 voces).

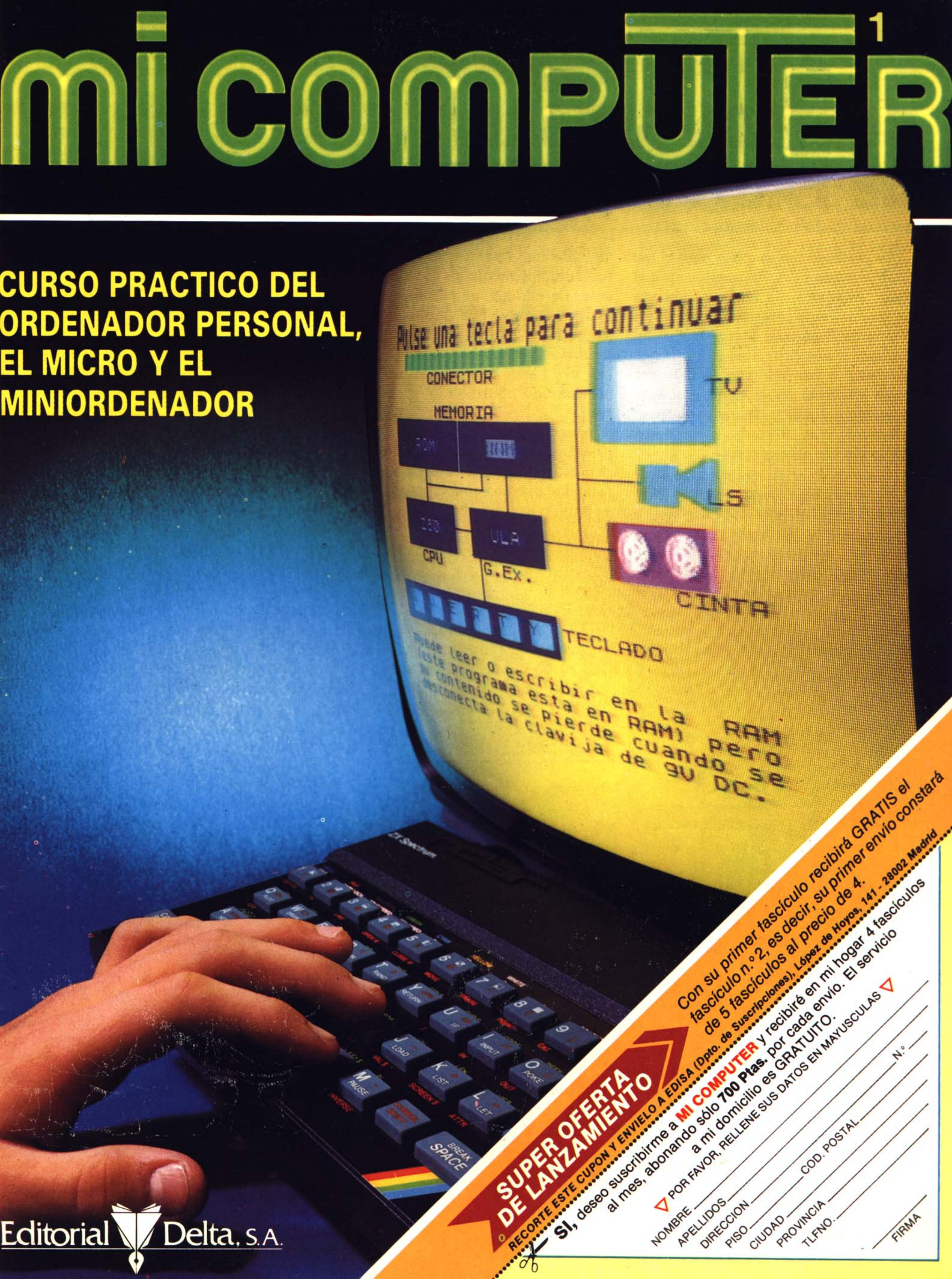
Novedad muy interesante es el reloj/calendario incorporado, alimentado a través de una pequeña batería que también alimenta algunas posiciones de RAM, con el que se podrá disponer de la hora, la fecha y alguna otra información de forma permanente, aún después de haber tenido el ordenador apagado.

Otra novedad la constituye la posibilidad de utilizar 32K de RAM como disco virtual (RAM DISK).

Pronto hablaremos con más detalle de estas nuevas máquinas, pero con las características que hemos mencionado (a un nivel de precios realmente sorprendente), auguramos un magnífico futuro para esta nueva generación.







mi computer
1
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
Pulse una tecla para continuar
CONECTOR
MEMORIA
RAM
TV
LS
CPU
ULA
G.EX.
CINTA
TECLADO
Puede leer o escribir en la RAM
(este programa esta en RAM) pero
su contenido se pierde cuando se
desconecta la clavija de 9V DC.
ZX Spectrum
SUPER OFERTA DE LANZAMIENTO
Con su primer fascículo recibirá GRATIS el fascículo n.º 2, es decir, su primer envío constará de 5 fascículos al precio de 4.
RECORTE ESTE CUPON Y ENVIELO A EDISA (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid
SI, deseo suscribirme a MI COMPUTER y recibiré en mi hogar 4 fascículos al mes, abonando sólo 700 Ptas. por cada envío. El servicio a mi domicilio es GRATUITO.
POR FAVOR, RELLENE SUS DATOS EN MAYUSCULAS
NOMBRE
APELLIDOS
DIRECCION
N.º
PISO
CIUDAD
COD. POSTAL
PROVINCIA
TLFNO.
FIRMA
Editorial Delta, S.A.

CONTROL GLOBAL DE FACTURACION IVA

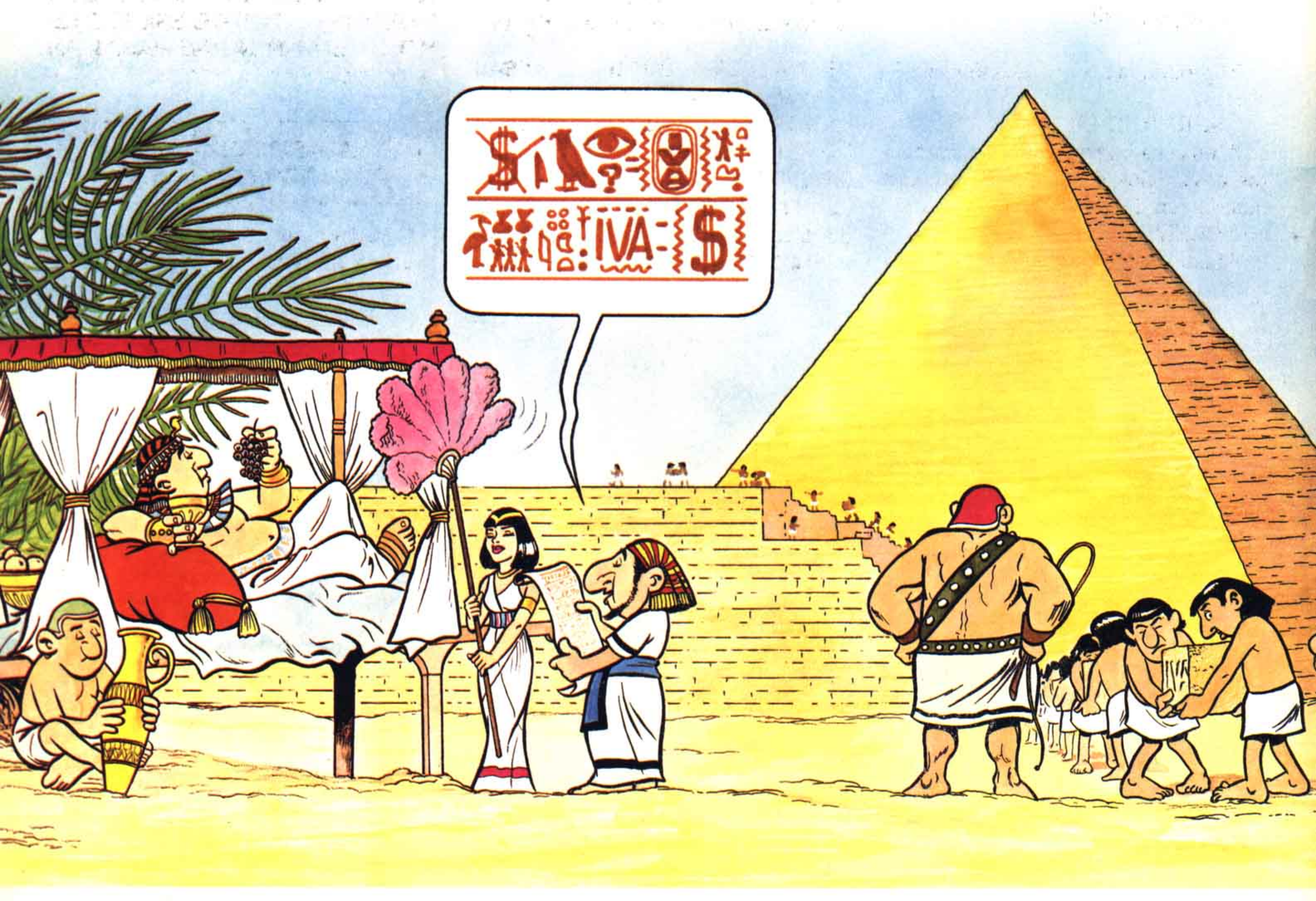
- HABLANDO DE IMPUESTOS
- SERVICIOS Y PRODUCTOS
- PROVEEDORES Y CLIENTES
- FACTURAS
- EL PROGRAMA

No pretendemos analizar aquí el tan controvertido IVA (Impuesto sobre el Valor Añadido). Su implantación nos ha llegado como consecuencia de la adhesión de España al Mercado Común Europeo y nos guste o no tenemos que aprender a convivir con él. Nuestra única intención es, al menos, facilitar las cosas a aquellos de nuestros lectores que se vean obligados a liquidar a Hacienda por este concepto.

El programa está escrito totalmente en BASIC para facilitar las cosas a aquellos que intenten adaptarlo o estudiar simplemente su composición y sacar de ello alguna enseñanza informática, objetivo éste que siempre nos proponemos. Se requiere una unidad de *diskette* e impresora de 80 columnas; por razones operativas, no hemos considerado oportuno la versión *cassette*.

Al redactar el programa se ha dado prioridad a una exposición lo más clara y estructurada posible y a una operatividad máxima aunque para ello haya habido necesidad de alargar el programa. Se ha procurado igualmente que el programa se encargue del máximo de validaciones para disminuir los errores de manejo por el usuario.

El objetivo del programa es llevar el registro de las facturas pagadas a nuestros proveedores por compras de bienes o servicios y también de las que nosotros emitamos a nuestros clientes. En cualquier momento podremos consultar una factura o un grupo de ellas, e incluso modificarlas o listarlas. También podremos conocer directamente



los datos necesarios para la liquidación de Hacienda.

Tanto la consulta como el listado pueden hacerse refiriéndonos a todos los apuntes de un fichero o de forma selectiva a algunos de ellos. Por ejemplo puede interesarnos conocer las facturas de un cierto proveedor, comprendidas entre dos fechas y cuyos importes superen un cierto valor, etc.

Por cada registro de documento contable pueden reflejarse los siguientes datos:

- Número de Factura
- Codificación y descripción de contenido (servicio o producto)
- Codificación y descripción del proveedor o cliente
- Fecha
- Importe (sin **IVA**)
- Importe del **IVA** con indicación de si es cargado o repercutido el % utilizado
- Codigo Auxiliar

El programa maneja tres tipos de ficheros:

* SERVIC.DAT

Almacena hasta 20 tipos de productos o servicios de uso frecuente con descripción de hasta 20 caracteres.

* CLIENT.DAT

Almacena hasta 60 tipos de clientes o proveedores de uso frecuente, con descripción de hasta 20 caracteres.

* Faann.DAT

Almacena datos de 100 facturas. Los caracteres (aa) corresponden a las dos últimas cifras del año y (nn) al número de orden del fichero (01-99). Por lo tanto F8602.DAT será el fichero que guarda las facturas número 101 a 200 del año 86.

En todo momento además del programa, residen en memoria los ficheros SERVICE.DAT, CLIENT.DAT y el fichero Faann.DAT correspondiente a la factura que en cada momento se requiera.

PUESTA EN MARCHA

Después de la pantalla de presentación, el ordenador nos solicíta el año (ej. 86) con el que vamos a trabajar, apareciendo a continuación un menú de operaciones como el de la figura 1.

```
01-INCLUIR / MODIF.FACTURAS
02-VISUALIZA.SELECTIVA FACT.
03-LISTADO SELECTIVO FACT.
04-RESUMENES LIQUID.HACIENDA
05-LISTADO SERVIC.IMPORTANT.
06-MODIFIC.SERVIC.IMPORTANT.
07-LISTADO CLIENT.IMPORTANT.
08-MODIFIC.CLIENT.IMPORTANT.
```

```
        FICHEROS EN USO

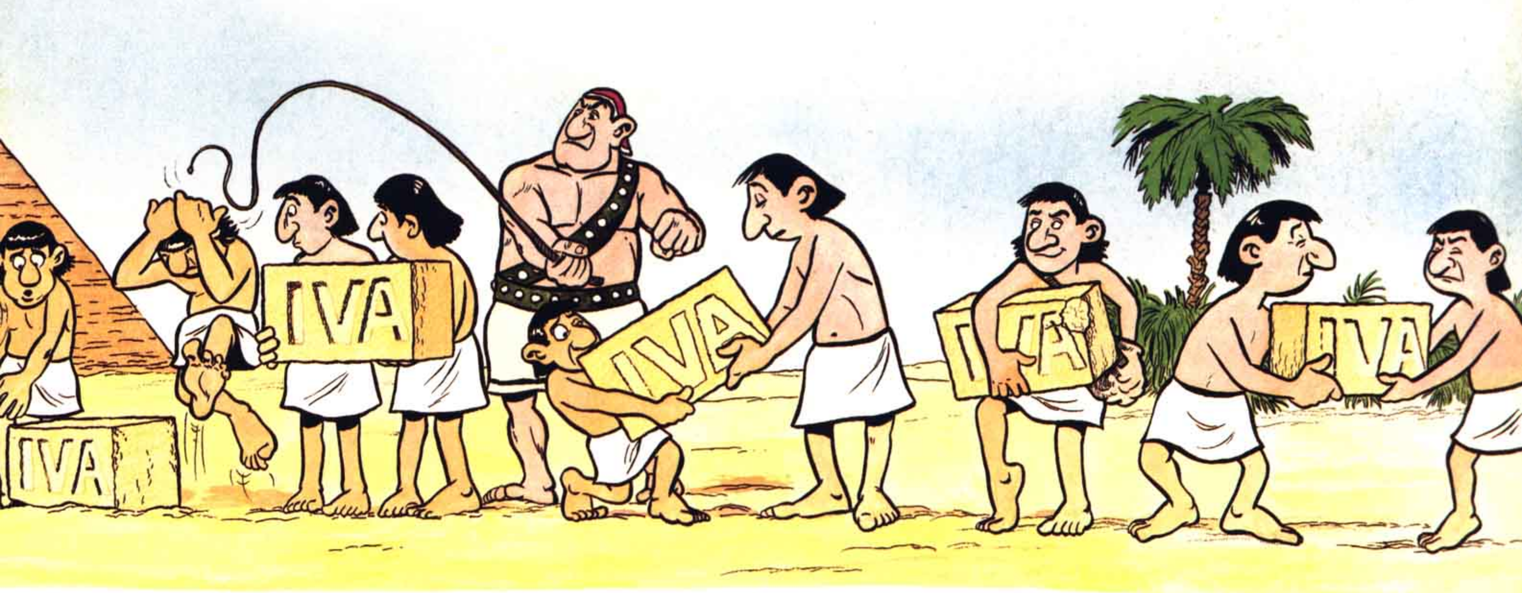
Faann  19aa   Fact .../...
```

La última línea nos permite ver en todo momento el fichero en uso así como el año y números de facturas a que corresponde. Si por ejemplo hemos elegido 1986, inicialmente aparecerá: F8600 1986 FACT. 0/0. Tan pronto nos pongamos a trabajar, por ejemplo en la factura número 275, la información cambiará a: F8603 1986 FACT. 201 / 300.

NOTA IMPORTANTE: SI POR ALGUNA CAUSA HAY QUE REINICIALIZAR EL RODAJE DEL PROGRAMA CUANDO SE ESTE TRABAJANDO CON UN FICHERO NUNCA HAGAS (RUN) SINO QUE DEBES REANUDAR CON (GO TO 1000) O DE OTRA MANERA PERDERAS TODOS LOS NUEVOS DATOS DE ESE FICHERO QUE AUN NO SE PASARON A DISCO.

TABLAS DE CODIGOS DE SERVICIOS / PRODUCTOS

Para mejorar las prestaciones se ha previsto codificar los servicios o pro-



ductos más importantes, en nuestro caso 20. De esta manera, posteriormente se podrá hacer un manejo selectivo de facturas.

Si una factura no está justificado clasificarla, simplemente no se le asignará ningún código o bien se le asignará el valor 00.

Con ayuda de la Opción 6 confeccionaremos o modificaremos la tabla correspondiente, que podremos consultar siempre que sea necesario mediante la opción 5 .

Por ejemplo las facturas de una empresa de construcción, independientemente de su descripción específica, podrían quedar clasificadas en: gastos generales, cemento, pinturas, alquiler de equipos de obra, modificaciones de tiendas, construcción de jardines, etc.

TABLAS DE CODIGOS DE PROVEEDORES / CLIENTES

Idéntica situación se nos presenta con los clientes o los proveedores.

Igualmente sólo se codificarán los tipos importantes. En nuestro caso se han previsto 60 tipos.

Con ayuda de la Opción 8 confeccionaremos la tabla correspondiente que podremos consultar mediante la opción 07.

En el mismo tipo de negocio anterior las facturas, además de indicar el proveedor / cliente podrían llevar el código correspondiente a dicho proveedor / cliente (o familias de) a efectos de manejo posterior.

INCLUSION O MODIFICACION DE FACTURAS

La inclusión de facturas pagadas, o cobradas, se realiza con ayuda de la Opción 1. La mecánica de inclusión está totalmente guiada por el programa y no ofrece problemas especiales.

El número inicial del orden de la factura lo fija el usuario. El ordenador fija automáticamente los siguientes, dentro de cada sesión. Si existe ya una factura creada con ese número, sus datos aparecen en pantalla, con lo cual se evitan cancelaciones o modificaciones involuntarias o, por el contrario, se puede aprovechar para hacerlas conscientemente.

Si en un cierto momento se rebasa la factura número 100 del fichero en curso, el programa guarda automáticamente el fichero en el *diskette* y carga el siguiente (o crea uno nuevo si no existe).

Al introducir el valor del **IVA** repercutido el ordenador calcula el % sobre el valor de la factura. En el caso de **IVA** (cargado) el programa solicita el % y calcula el valor correspondiente.

El campo de Cod. Aux. (hasta tres caracteres) puede utilizarse según las necesidades.

En el momento de introducir el código de servicio o cliente puede consultarse la tabla correspondiente si se desea (* seguido de RETURN). Una vez fijado el código no es necesario introducir el texto correspondiente en la tabla; lo cual resulta cómodo y evita errores. Como ya se ha indicado, si no se trata de un producto o cliente predeterminado, puede codificarse tecleando la información correspondiente. Al final de cada sesión de trabajo, el programa se encarga de guardar automáticamente, en *diskette*, el fichero que se está manejando.

BUSQUEDA SELECTIVA

Sin necesidad de acudir a un listado en papel podemos conocer y visualizar en pantalla, una a una, todas las facturas del fichero que cumplen una cierta condición. Por ejemplo aquellas «pagadas» entre «tal y tal fecha», de un «tipo de servicio», y/o un «tipo de proveedor» determinado. Igualmente pueden fijarse los límites (mínimo, máximo o ambos) del importe de la factura y del IVA.

No prefijando ningun límite, lógicamente se visualizan todas las facturas archivadas.

LISTADO SELECTIVO

Aquellos que dispongan de una impresora de 80 columnas conectada a su **MSX** pueden aprovechar la Opción 3 para obtener un listado en papel con los datos análogos a los ofrecidos por la Opción 2.

Al final del listado aparecerán, también en papel, los criterios de selección y el resumen para Hacienda de todas las facturas listadas.

RESUMEN PARA LIQUIDACION A HACIENDA

En definitiva, el objetivo del programa radica en llevar un control de nuestra actividad, a través de las facturas emitidas o pagadas, para poder conocer con exactitud como va nuestro balance de **IVA** (cobrado «C»y pagado «R») a efectos de nuestras relaciones con Hacienda. Con ayuda de la Opción 4 podemos ver como va en todo momento este balance.

El programa nos pide dos fechas y explora todas las facturas archivadas entre ellas («*» si queremos todas las facturas independientemente de su ficha), presentando en pantalla los resultados correspondientes como se indica en el ejemplo de la figura 2.

```
FACTURAS        de :   20/05/86
                a  :   29/05/86

TIPO     FAC. IMPORTE      IVA
----     ---- -------      ---

IVA "C" 5     1361679   136145
IVA "R" 3     4367033    89640
        ---   -------   ------
        8     5728712    46505
```

El programa considera que no existen más facturas cuando el importe de la última explorada es 0. Esto puede dar lugar a errores si, por descuido, nos dejamos alguna factura «en blanco». Verificar siempre mediante la opción 2.

```
10 REM PROGRAMA CGFIVA MSX
20 REM Control General
   Facturacion IVA
30 REM ****************
40 REM *** MOLISOFT ***
50 REM ****************
51 CLEAR 13000
```

```
52 COLOR 1,15,15: KEY OFF:
   SCREEN 0:WIDTH 32:CLS
53 DIM SR$(20),CL$(60),FA$
   (100),DT$(10),D1$(10),D2$
   (10)
54 FOR I=1 TO 20:SR$(I)=
   SPACE$(20):NEXT I
55 FOR I=1 TO 60:CL$(I)=
   SPACE$(20):NEXT I
56 FOR I=1 TO 100:FA$(I)=
   SPACE$(75):NEXT I
57 DR$="A:":NR$="00":AR$="86"
   :TP$=".DAT"
59 DEF FNFR$="F"+AR$+NR$:DEF
   FNFS$="F"+AR$+NS$
62 NR=0:NS=0:NF=0:RF=0:NL=0:
   NM=9900
65 DEF FNST$(VL)=MID$(STR$
   (VL),2)
67 DEF FNVL$(N$,L)=STRING$
   ((L-LEN(N$)),"0")+N$
158 LOCATE 0,5:PRINT STRING$
    (32,"*")
160 LOCATE 1,8:PRINT"CONTROL
    GENERAL DE FACTURACION"
162 LOCATE 13,11:PRINT
    "I V A"
163 LOCATE 11,14:PRINT"INPUT
    MSX"
166 LOCATE 0,17:PRINT STRING$
    (32,"*")
167 TIME=0
168 IF TIME <150 THEN 168
170 GOSUB 9020: INPUT"A%O "
    ;AR$:IF LEN(AR$)<>2 THEN
    170
200 ON ERROR GOTO 9900
210 GOSUB 9410
220 GOSUB 9450
1000 '
1001 REM MENU
1002 '
1005 CLS:PRINT"CONTROL GLOBAL
     DE FACTURACION",STRING$
     (29,"="):PRINT
1015 PRINT"01 - INCLUIR /
     MODIF. FACTURAS"
1045 PRINT"02 - VISUALIZA.
     SELECTIVA FACT."
1055 PRINT"03 - LISTADO
     SELECTIVO FACTURAS"
1085 PRINT"04 - RESUMENES
     LIQUID. HACIENDA"
1087 PRINT"05 - LISTADO
     SERVIC. IMPORTANT"
1089 PRINT"06 - MODIFIC.
     SERVIC. IMPORTANT"
1091 PRINT"07 - LISTADO
     CLIENT. IMPORTANT"
1093 PRINT"08 - MODIFIC.
     CLIENT. IMPORTANT"
1106 GOSUB 9025
1107 PRINT:PRINT:PRINT TAB(9)
     ; "FICHERO EN USO ":
     PRINT
1108 PRINT FNFR$;TAB(8);"19"+
     AR$;" FACT. ";PF;"/";UF
1109 PRINT: PRINT"OPCION ?"
1110 P$=INKEY$:IF P$="" THEN
     1110
1115 IF VAL(P$)<1 OR VAL(P$)>
     9 THEN 1110
1120 P=VAL(P$):ON P GOSUB4500
     ,5000,5500,6500,6100,
     6200,6300,6400K
1125 GOTO 1000
3425 '
3426 REM PARTES FACTURAS
3427 '
3430 CLS:LOCATE 0,1:PRINT
     "FACTURA No.:";TAB(10);
     STRING$(20," "):LOCATE
     0,2:PRINT STRING$
     (11,"-")
3432 LOCATE 0,4:PRINT"1  COD.
     SER.:"
3433 PRINT"2  SERVICIO:"
3434 PRINT"3  COD.CLI.:"
3435 PRINT"4  CLIENTE :"
3437 PRINT"5  FECHA   :"
3439 PRINT"6  IMP.NETO:"
3441 PRINT"7  IVA C/R :"
3443 PRINT"8  IMP.IVA :"
3445 PRINT"9  COD.AUX.:"
3446 RETURN
3447 '
3448 REM DATOS ALTAS/MODIF
3449 '
3500 LOCATE 12,1:PRINT NF
3505 GOSUB 9020:INPUT"COD.SER
     .(max.2c)  *=CONSULTA";
     DT$(1)
3506 IF LEN(DT$(1))>2 THEN
     3505
3507 IF DT$(1)="*" THEN GOSUB
     6100:GOSUB 3430:DT$(2)=
     SR$(VAL(DT$(1))):GOTO
     3500
3508 DT$(2)=SR$(VAL(DT$(1)))
3510 LOCATE 12,4:PRINT STRING
     $(20," "): LOCATE 12,4:
     PRINT DT$(1)
3515 GOSUB 9020:INPUT
     "SERVICIO";DT$(2):IF LEN
     (DT$(2))>20 THEN 3515
3520 LOCATE 12,5:PRINT SPACE$
     (20): LOCATE 12,5:PRINT
     DT$(2)
3525 GOSUB 9020:INPUT"COD.CLI
     .(max. 2c) *=CONSULTA";
     DT$(3)
3526 IF LEN(DT$(3))>2  THEN
     3525
3527 IF DT$(3)="*" THEN GOSUB
     6300:GOSUB 3430:LOCATE
     12,1:PRINT NF:LOCATE
     12,4:PRINT DT$(1):GOTO
     3520:DT$(4)=CL$(VAL(DT$
     (3)))
3528 DT$(4)=CL$(VAL(DT$(3)))
3530 LOCATE 12,6:PRINT SPACE$
     (20): LOCATE 12,6:PRINT
     DT$(3)
3532 GOSUB 9020:INPUT
     "CLIENTES";DT$(4):IF LEN
     (DT$(4))>20 THEN 3532
3534 LOCATE 12,7:PRINT SPACE$
     (20): LOCATE 12,7:PRINT
     DT$(4)
3535 GOSUB 9020:INPUT"FECHA
     DD/MM ";DT$(5):IF LEN
     (DT$(5))<>5 THEN 3535
3536 V1=VAL(MID$(DT$(5),1,2))
     :V2=VAL(MID$(DT$(5),4,2)
     ):IF V1<1 OR V1>31 OR V2
     <0 OR V2>12  OR MID$(DT$
     (5),3,1)<>"/" THEN 3535
3540 LOCATE 12,8:PRINT SPACE$
     (20): LOCATE 12,8:PRINT
     DT$(5)+"/"+AR$
3545 GOSUB 9020:INPUT"IMPORTE
     NETO ( SIN IVA )";DT$(6)
     :IF LEN(DT$(6))>7 THEN
     3545
3550 LOCATE 12,9:PRINT SPACE$
     (20): LOCATE 12,9:PRINT
     DT$(6)
3555 GOSUB 9020:INPUT"IVA
     COBRADO/REPERCUTIDO?
     (C/R)";DT$(7):IF DT$(7)<
     >"C" AND DT$(7)<>"R"
     THEN 3555
3557 LOCATE 12,10:PRINTSPACE$
     (20):LOCATE 12,10:PRINT
     DT$(7)
```

```
3560 IF DT$(7)="C" THENLOCATE
     0,20:PRINT SPACE$(64):
     LOCATE 0,20:INPUT"%IVA
     (Ej. 06.0) ";DT$(8):IF
     LEN(DT$(8))<>4 OR MID$
     (DT$(8),3,1)<>"." THEN
     3560
3562 IF DT$(7)="C" THEN V1=
     VAL(DT$(8)):V2=VAL(DT$
     (6)):V=(V1*V2)/100+.5:V=
     INT(V):DT$(9)=FNST$(V):
     LOCATE 12,11:PRINTSPACE$
     (20): LOCATE 12,11:PRINT
     DT$(8);" %  ";DT$(9);
     " PTS"
3565 IF DT$(7)="R" THENLOCATE
     0,20:PRINT SPACE$(64):
     LOCATE 0,20:INPUT
     "IMPORTE IVA ";DT$(9):IF
     LEN(DT$(9))>7 THEN 3565
3570 IF DT$(7)="R" THENV1=VAL
     (DT$(6)):V2=VAL(DT$(9)):
     VL=(INT((V2/V1+5E-04)*
     1000))/10:DT$(8)=FNST$
     (VL):LOCATE 12,11:PRINT
     SPACE$(20): LOCATE 12,11
     :PRINT DT$(9);" PTS  ";
     DT$(8);" %"
3575 GOSUB 9020:INPUT"COD.
     AUX.";DT$(10):IF LEN
     (DT$(10))>3 THEN 3575
3580 LOCATE 12,12:PRINTSPACE$
     (20): LOCATE 12,12:PRINT
     DT$(10)
3585 RETURN
4500 '
4505 REM INCLUSIONES/
     MODIFICACIONES
4507 '
4510 CLS:NF=1:PRINT:INPUT
     "FACTURA No.";NF:IF NF<0
     OR NF>NM  THEN 4510
4520 NS=INT((NF-1)/100)+1:NS$
     =FNST$(NS):NS$=FNVL$
     (NS$,2):RF=(NS-1)*100
     :NL=NF-RF
4525 IF NS$=NR$  THEN 4550
4530 ON ERROR GOTO 9900
4535 IF NR$<>"00" THEN GOSUB
     9710
4537 FOR I=1 TO 100:FA$(I)=
     STRING$(75," "):NEXT I
4540 GOSUB 9810
4545 NR$=NS$
4550 GOSUB 3430
4552 GOSUB 7574:GOSUB 7610
4555 GOSUB 3500
4560 LOCATE 0,20:PRINTSTRING$
     (64," "):LOCATE 0,20:
     PRINT" MODIFICAR ? S/N "

4565 P$=INKEY$:IF P$="" THEN
     4565
4570 IF P$="S" OR P$="s" THEN
     4555
4572 DT$(0)=FNST$(NF):DT$(0)=
     FNVL$(DT$(0),4):DT$(1)=
     FNVL$(DT$(1),2):DT$(3)=
     FNVL$(DT$(3),2)
4573 FA$(NL)=STRING$(75," ")
4574 MID$(FA$(NL),1,4)
     =DT$(0)
4575 MID$(FA$(NL),5,2)
     =DT$(1)
4576 MID$(FA$(NL),7,20)
     =DT$(2)
4578 MID$(FA$(NL),27,2)
     =DT$(3)
4579 MID$(FA$(NL),29,20)
     =DT$(4)
4580 MID$(FA$(NL),49,5)
     =DT$(5)
4581 MID$(FA$(NL),54,7)
     =DT$(6)
4582 MID$(FA$(NL),61,1)
     =DT$(7)
4583 MID$(FA$(NL),62,4)
     =DT$(8)
4584 MID$(FA$(NL),66,7)
     =DT$(9)
4585 MID$(FA$(NL),73,3)
     =DT$(10)
4590 LOCATE 0,20:PRINTSTRING$
     (64," "):LOCATE 0,20:
     PRINT" CONTINUAR  S/N "
4595 P$=INKEY$:IF P$="" THEN
     4595
4598 IF P$="N" OR P$="n"THEN
     4700
4600 NF=NF+1:GOTO 4520
4700 GOSUB 9710:GOTO 1000
5000 '
5002 REM BUSQUEDA SELECTIVA
5005 '
5007 CLS
5008 FOR I=0 TO 10:D1$(I)="*"
     :NEXT I
5009 FOR I=0 TO 10:D2$(I)="*"
     :NEXT I
5010 GOSUB 9020:INPUT"PRIMERA
     FACTURA  *=TODAS ";
     D1$(0):IF D1$(0)="*"
     THEN D1$(0)="1"
5015 GOSUB 9020:INPUT"ULTIMA
     FACTURA *=TODAS";D2$(0)
5020 GOSUB 9020:INPUT"CODIGO
     SERVICIO *=TODOS ";
     D1$(1)
5025 GOSUB 9020:INPUT"CODIGO
     CLIENTE *=TODOS";D1$(3)
5030 GOSUB 9020:INPUT"FECHA
     MAS BAJA *=TODAS ";
     D1$(5)
5035 GOSUB 9020:INPUT"FECHA
     MAS ALTA *=TODAS ";
     D2$(5)
5040 GOSUB 9020:INPUT"IMPORTE
     MINIMO *=TODOS ";D1$(6)
5045 GOSUB 9020:INPUT"IMPORTE
     MAXIMO *=TODOS ";D2$(6)
5050 GOSUB 9020:INPUT"TIPO
     IVA C/R  *=TODOS ";
     D1$(7)
5055 GOSUB 9020:INPUT"% IVA
     MIN. (Ej.12.0) *=TODOS";
     D1$(8)
5060 GOSUB 9020:INPUT"% IVA
     MAX. (Ej.12.0) *=TODOS";
     D2$(8)
5070 GOSUB 9020:INPUT"COD.
     AUX.  *=TODOS ";D1$(10)
5110 NF=VAL(D1$(0))
5120 NS=INT((NF-1)/100)+1:NS$
     =FNST$(NS):NS$=FNVL$
     (NS$,2):RF=(NS-1)*100:NL
     =NF-RF
5125 IF NS$=NR$  THEN 5147
5130 ON ERROR GOTO 9900
5135 IF NR$<>"00" THEN GOSUB
     9710
5140 GOSUB 9810
5145 NR$=NS$
5147 IF P=2 THEN GOSUB 3430
5150 GOSUB 7574
5155 IF (P=3 OR P=4) AND
     VAL(DT$(6))=0 THEN 5200
5156 IF DT$(7)="C" THEN R1=R1
     +1:R2=R2+VAL(DT$(6)):R3=
     R3+VAL(DT$(9))
5157 IF DT$(7)="R" THEN R4=R4
     +1:R5=R5+VAL(DT$(6)):R6=
     R6+VAL(DT$(9))
5172 IF D1$(1)="*" THEN 5174
     ELSE IF VAL(DT$(1))<>VAL
     (D1$(1)) THEN 5199
```

```
5174 IF D1$(3)="*" THEN 5176
     ELSE IF VAL(DT$(3))<>VAL
     (D1$(3)) THEN 5199
5176 IF D1$(5)="*" THEN 5178
     ELSE IF DT$(5)<D1$(5))
     OR DT$(5)>D2$(5) THEN
     5199
5178 IF D1$(6)="*" THEN 5180
     ELSE IF VAL(DT$(6))<VAL
     (D1$(6)) OR VAL(DT$(6))>
     VAL(D2$(6))THEN 5199
5180 IF D1$(7)="*" THEN 5182
     ELSE IF DT$(7)<>D1$(7)
     THEN 5199
5182 IF D1$(9)="*" THEN 5184
     ELSE IF VAL(DT$(9))<VAL
     (D1$(9)) OR VAL(DT$(9))>
     VAL(D2$(9))THEN 5199
5184 IF D1$(10)="*" THEN 5185
     ELSE IF DT$(10)<>D1$(10)
     THEN 5199
5185 IF  P=2 THEN GOSUB 7600
5187 IF  P=3 THEN GOSUB 8600
5190 IF P=4 THEN CLS:LOCATE
     0,10:PRINT"UN MOMENTO .
     . .":GOTO 5199
5191 LOCATE 0,20:PRINT SPACE$
     (64):LOCATE 0,20:PRINT
     " CONTINUAR  S/N "
5195 P$=INKEY$:IF P$="" THEN
     5195
5198 IF P$="N" OR P$="n"THEN
     GOTO 1000
5199 NF=NF+1:IF VAL(D2$(0))<>
     0 AND NF> VAL(D2$(0))
     THEN 5200 ELSE 5120
5200 IF P=4 THEN 6600
5205 IF P=3 THEN 8650 ELSE
     LOCATE 0,20:PRINT"FIN
     FACTURAS
     (PULSE RETURN)"
5210 P$=INKEY$:IF P$="" THEN
     5210
5310 GOTO 1000
5500 '
5510 REM LISTADO SELECTIVO
5512 '
5520 RG=1:PG=1
5530 CLS:LOCATE 7,10:PRINT
     "CONECTE LAS IMPRESORA":
     TIME=0
5535 IF TIME<150 THEN 5535
5540 GOTO 5000
6000 '
6005 REM  SERVICIOS/CLIENTES
     IMPORTANTES
6100 CLS:PRINT "SERVICIOS
     IMPORTANTES :":PRINT
     STRING$(21,"=")
6110 FOR I=1 TO 20:PRINT I;
     TAB(4);SR$(I):NEXT I
6117  PRINT "PULSE CUALQUIER
     TECLA"
6122 IF INKEY$=""THEN 6122
     ELSE RETURN
6200 CLS:INPUT"SERVICIO No.
     (0=FINAL)";N:IF N<0 OR
     N>20 THEN 6200
6210 IF N=0 THEN  GOSUB 9610:
     RETURN
6220 PRINT:PRINT"ACTUAL:";
     TAB(8);SR$(N)
6230 PRINT:INPUT"NUEVO NOMBRE
     (*=CONSERVAR)  ";N$:IF
     LEN(N$)>20 THEN 6230
6235 PRINT"C";TAB(5);R1;TAB
     (10);R2;TAB(20);R3
6240 IF N$="*" THEN 6200 ELSE
     SR$(N)=N$:GOTO 6200
6300 FOR L=0 TO 2
6305 CLS:PRINT "CLIENTES
     IMPORTANTES :":PRINT
     STRING$(20,"=")
6310 FOR I=1 TO 20:PRINT
     (20*L+I);TAB(4);CL$
     (20*L+I):NEXT I
6317  PRINT "PULSE CUALQUIER
     TECLA"
6322 IF INKEY$=""THEN 6322
     ELSE NEXT L:RETURN
6400 CLS:INPUT"CLIENTE No.
     (0=FINAL)";N:IF N<0 OR
     N>60 THEN 6400
6410 IF N=0 THEN  GOSUB 9510:
     RETURN
6420 PRINT:PRINT"ACTUAL:";TAB
     (8);CL$(N)
6430 PRINT:INPUT"NUEVO NOMBRE
     (*=CONSERVAR)  ";N$:IF
     LEN(N$)>20 THEN 6430
6440 IF N$="*" THEN 6400 ELSE
     CL$(N)=N$:GOTO 6400
6500 '
6505 REM LIQUIDACION
     HACIENDA
6506 '
6510 D1$(0)="1":D1$(5)="*":
     D2$(5)="*":R1=0:R2=0:
     R3=0:R4=0:R5=0:R6=0
6530 CLS:GOSUB 9020:INPUT
     "FECHA INF. (DD/MM)
     (* =TODAS)";D1$(5)
6540 GOSUB 9020:INPUT"FECHA
     SUP. (DD/MM) (* =TODAS)"
     ;D2$(5):GOTO 5110
6550 GOTO 5110
6600  CLS:PRINT"  RESUMEN
     LIQUIDACION HACIENDA": P
     RINT STRING$(32,"="): PR
     INT
6610 PRINT"FACTURAS DE :";D1$
     (5)+"/"+AR$+"  A  "+D2$
     (5)+"/"+AR$:PRINT
6630 PRINT"TIPO  FACTS.
     IMPORTE  IVA":PRINT
6635 PRINT"IVA C";TAB(6);R1;
     TAB(12);R2;TAB(22);R3
6640 PRINT"IVA R";TAB(6);R4;
     TAB(12);R5;TAB(22);R6
6650 L1=R1+R4:L2=R2+R5
     :L3=R3-R6
6655 PRINT "        ====  ====
     ====  ========"
6660 PRINT"NETO";TAB(6);L1;
     TAB(12);L2;TAB(22);L3
6900 LOCATEO,20:PRINT" RETURN
     PARA SEGUIR":GOTO 5210
7500 '
7510 REM LECT. CAMPOS
     FACTURAS
7515 '
7574 DT$(0)= MID$
     (FA$(NL),1,4)
7575 DT$(1)= MID$
     (FA$(NL),5,2)
7576 DT$(2)=MID$
     (FA$(NL),7,20)
7578 DT$(3)=MID$
     (FA$(NL),27,2)
7579 DT$(4)=MID$
     (FA$(NL),29,20)
7580 DT$(5)=MID$
     (FA$(NL),49,5)
7581 DT$(6)=MID$
     (FA$(NL),54,7)
7582 DT$(7)=MID$
     (FA$(NL),61,1)
7583 DT$(8)=MID$
     (FA$(NL),62,4)
7584 DT$(9)=MID$
     (FA$(NL),66,7)
7585 DT$(10)=MID$
     (FA$(NL),73,3)
7590 RETURN
7595 '
```

```
7596 REM VISUALIZACION
     VALORES CAMPOS
7597 '
7600 LOCATE 12,1:PRINT SPACE$
     (20):LOCATE 12,1:PRINT
     DT$(0)
7610 LOCATE 12,4:PRINT SPACE$
     (20): LOCATE 12,4:PRINT
     DT$(1)
7620 LOCATE 12,5:PRINT SPACE$
     (20): LOCATE 12,5:PRINT
     DT$(2)
7630 LOCATE 12,6:PRINT SPACE$
     (20): LOCATE 12,6:PRINT
     DT$(3)
7640 LOCATE 12,7:PRINT SPACE$
     (20): LOCATE 12,7:PRINT
     DT$(4)
7650 LOCATE 12,8:PRINT SPACE$
     (20): LOCATE 12,8:PRINT
     DT$(5)+"/"+AR$
7660 LOCATE 12,9:PRINT SPACE$
     (20): LOCATE 12,9:PRINT
     DT$(6)
7670 LOCATE 12,10:PRINT SPACE
     $(20):LOCATE 12,10:PRINT
     DT$(7)
7680 LOCATE 12,11:PRINT SPACE
     $(20):LOCATE 12,11:PRINT
     DT$(8);" %  ";DT$(9);" P
     TS"
7690 LOCATE 12,12:PRINT SPACE
     $(20):LOCATE 12,12:PRINT
     DT$(10)
7700 RETURN
8600 IF RG=1 THEN LPRINT:
     LPRINT:LPRINT:LPRINT:
     LPRINT  TAB(70);"PAG. ";
     PG:LPRINT:LPRINT"LISTADO
     SELECTIVO DE FACTURAS":
     LPRINT:LPRINT:LPRINT"NF
     CS SERV. / PROD. CC
     CLIENTE/PROVEEDOR FECHA
     IMPOR. T  %    IVA   AUX"
     :LPRINT:RG=10
8610 LPRINT DT$(0)+DT$(1)+" "
     +DT$(2)+DT$(3)+" "+DT$
     (4)+DT$(5)+" "+DT$(6)+DT
     $(7)+" "+DT$(8)+" "+DT$
     (9)+DT$(10)
8620 RG=RG+1:IF RG>62 THEN RG
     =1:PG=PG+1:LPRINT:LPRINT
     :LPRINT:LPRINT:LPRINT
8630 RETURN
8650 DF=(66-RG)+5
8655 FOR I=1 TO DF:LPRINT:
     NEXT I
8660 LPRINT TAB(70);"PAG. ";
     PG:LPRINT:LPRINT
     "LISTADO SELECTIVO DE
     FACTURAS":LPRINT:LPRINT
     :LPRINT"CRITERIOS DE
     SELECCION :":LPRINT
8665 LPRINT"No.FACTURAS :";
     D1$(0);" A ":D2$(0)
8667 LPRINT"COD. SERV.   :";
     D1$(1)
8669 LPRINT"COD. CLIENT.:";
     D1$(3)
8671 LPRINT"FECHAS        :";
     D1$(5)+"/"+AR$;" A ":D2$
```

```
     (5)+"/"+AR$
8673 LPRINT"IMP.(S.IVA) :";
     D1$(6);" A ":D2$(6)
8675 LPRINT"TIPO C/R     :";
     D1$(7)
8678 LPRINT" % IVA       :";
     D1$(8);" A ":D2$(8)
8681 LPRINT"IMPORTE IVA :";
     D1$(9);" A ":D2$(9)
8683 LPRINT"COD. AUX.    :";
     D1$(10)
8690 LPRINT:LPRINT"  RESUMEN
     LIQUIDACION HACIENDA":
     LPRINT STRING$(32,"="):
     LPRINT
8692 LPRINT"FACTURAS DE :";
     D1$(5)+"/"+AR$+"  A  "
     +D2$(5)+"/"+AR$:LPRINT
8694 LPRINT"TIPO  FACTS.
     IMPORTE  IVA":LPRINT
8696 LPRINT"IVA C";TAB(6);R1;
     TAB(12);R2;TAB(22);R3
8698 LPRINT"IVA R";TAB(6);R4;
     TAB(12);R5;TAB(22);R6
8700 L1=R1-R4:L2=R2+R5:L3=
     R3-R6
8702  LPRINT "        ==== ==
     ======  ========"
8704  LPRINT"NETO";TAB(6);L1;
     TAB(12);L2;TAB(22);L3
8706 LOCATE 0,20:PRINT
     "RETURN PARA SEGUIR":
     GOTO 5210
9000 '
9010 REM RUTINAS AUXILIARES
9011 '
9020 LOCATE 0,20:PRINTSTRING$
     (64," "):LOCATE 0,20:
     RETURN
9025 REM CALCULO DATOS
     FICHERO
9026 '
9030 UF=100*VAL(NR$):IF UF=0
     THEN PF=0 ELSE PF=UF-99
9035 RETURN
9410 OPEN DR$+"CLIENT.DAT"
     FOR INPUT AS #1
9420 FOR I=0 TO 60:INPUT #1,
     CL$(I):NEXT I
9430 CLOSE #1: RETURN
9450 OPEN DR$+"SERVIC.DAT"
     FOR INPUT AS #1
9460 FOR I=0 TO 20:INPUT #1,
     SR$(I):NEXT I
9470 CLOSE #1: RETURN
9510 OPEN DR$+"CLIENT.DAT"
     FOR OUTPUT AS #1
9520 FOR I=0 TO 60:PRINT #1,
     CL$(I):NEXT I
9530 CLOSE #1: RETURN
9610 OPEN DR$+"SERVIC.DAT"
     FOR OUTPUT AS #1
9620 FOR I=0 TO 20:PRINT #1,
     SR$(I):NEXT I
9630 CLOSE #1: RETURN
9710 OPEN DR$+FNFR$+TP$ FOR
     OUTPUT AS #1
9720 FOR I=0 TO 100:PRINT #1,
     FA$(I):NEXT I
9730 CLOSE #1: RETURN
9810 OPEN DR$+FNFS$+TP$ FOR
     INPUT AS #1
9820 FOR I=0 TO 100:INPUT #1,
     FA$(I):NEXT I
9830 CLOSE #1: RETURN
9900 REM MANEJO DE ERRORES"
9905 PRINT "ERROR ";ERR;
     " LINEA ";ERL
9910 IF ERL=9810 AND ERR=53
     THEN RESUME 4545
9920 IF ERL=9410 AND ERR=53
     THEN RESUME 220
9930 IF ERL=9450 AND ERR=53
     THEN RESUME 1000
```



CREACION DE FUNCIONES DE USUARIO

- COMO DEFINIR TUS PROPIAS FUNCIONES
- USO DE LAS FUNCIONES DE ENTRADA
- FUNCIONES INTERESANTES

En BASIC no hay funciones para cosas como cubos, interés compuesto o letras mayúsculas. Pero en muchos ordenadores se pueden crear funciones especiales cuando se necesitan.

Tu ordenador es capaz de hacer muchas cosas —en especial las relacionadas con los cálculos matemáticos— con mucha más rapidez que tu. Pero no puede hacer más que algo que le indiques previamente: o lo dices tu tecleando un programa, o se lo dice un programa que cargues (con LOAD) en tu ordenador.

El lenguaje más común que la gente suele utilizar para programar es el BASIC, el que se inicializa automáticamente al encender el equipo. Tu ordenador dispone de varias funciones incorporadas en el BASIC estandar: son funciones como SIN, COS, SQR y otras muchas. Cada una de ellas está definida de tal forma que tu ordenador sabe cómo reconocer el comando y realizar la operación adecuada sobre un valor dado, calculando su seno, su coseno, su raíz cuadrada o lo que sea.

FUNCIONES EN BASIC

Todo esto está muy bien en la medida en que las funciones que quieras usar en tu programa sean también **palabras clave** del BASIC de tu ordenador. Pero hay muchas funciones comunes que no son parte del BASIC estandar; por ejemplo, no hay ninguna función que se llame CUBO, que te calcule automáticamente el valor de X*X*X. Si no está disponible una función que vayas a utilizar muy a menudo, tienes varias opciones. Puedes buscar la forma de evitar el uso de esa función en tu programa; también puedes añadir una subrutina que calcule la función deseada; y, lo más elegante de todo, puedes emplear la facilidad de las funciones definibles por el usuario que tiene tu ordenador.

Literalmente las funciones definibles por el usuario te permiten configurar el BASIC a la medida de las necesidades de un programa particular. La forma básica del comando que define la función es la siguiente:

DEF FN a(x,y) = ... (lo que quieras que haga la función).

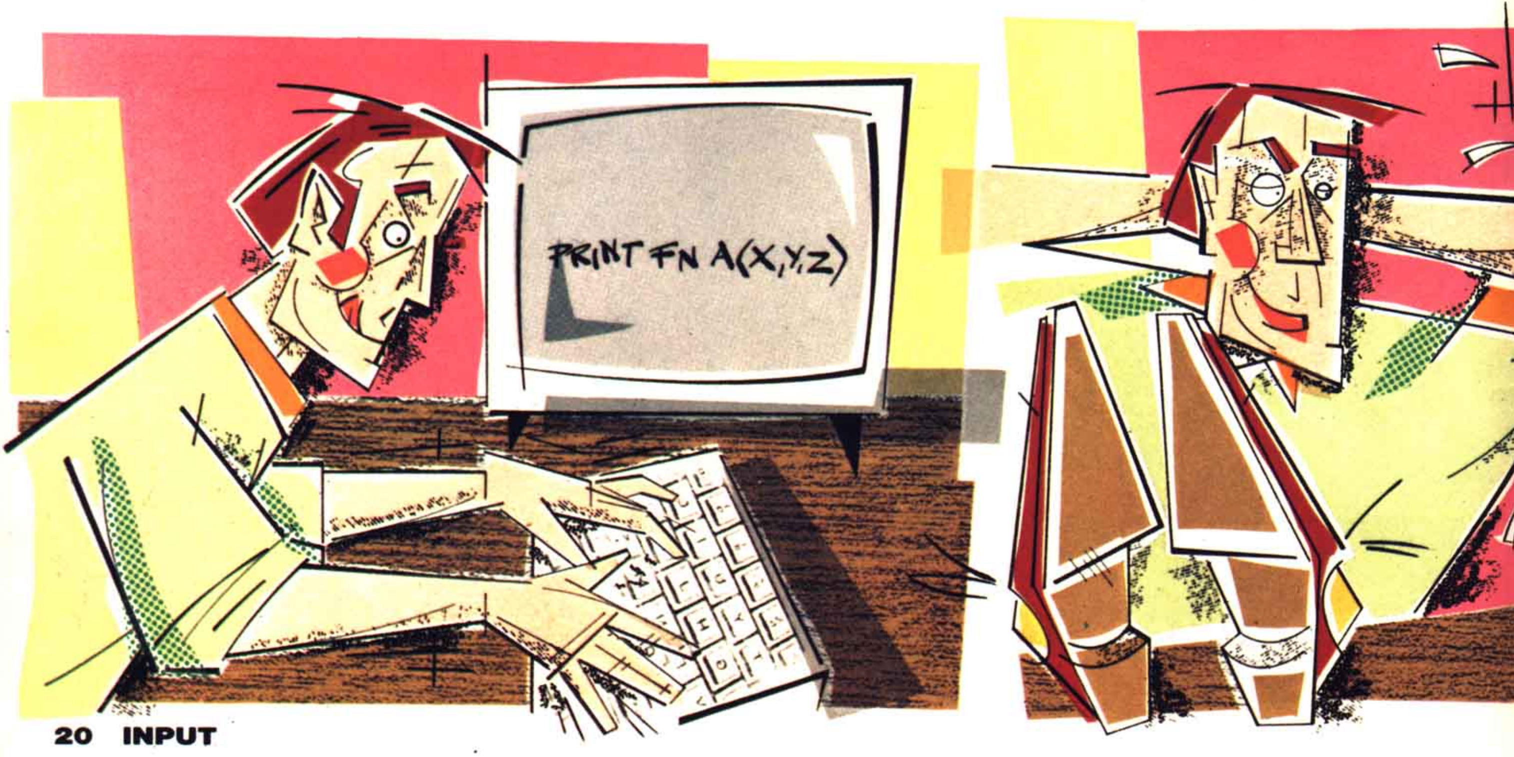
Las letras entre paréntesis son los parámetros de la función.

Para ver cómo funciona el principio general en un programa real, teclea y ejecuta el siguiente ejemplo. En él se define la sencilla función que te da el cubo de un número.

```
10 DEF FNC(X)=X*X*X
20 PRINT "Numero","Cubo"
30 FOR A=1 TO 10
40 PRINT A,FNC(A)
50 NEXT
```

LLAMADA A UNA FUNCION

Es muy fácil llamar a una función; para ello, cada vez que precises el re-



sultado númerico del cálculo de dicha función, teclea la palabra clave FN y a continuación el nombre de la función deseada. Esto es lo que se hace en la línea 40 del programa. En este ejemplo el programa utiliza FNC(A), donde A es el número cuyo cubo has de calcular. Si quieres que tu programa utilice esta función, no tienes más que usar FNC(A) en todos los sitios en que en otro caso pondrías A*A*A.

Este mismo principio se puede aplicar para cualquier otro uso de funciones definidas por el usuario; de hecho, la llamada a funciones se puede considerar una forma de abreviatura en los cálculos.

Para una función tan sencilla como la de este ejemplo, podría pensarse que es más rápido utilizar la función de elevación a potencias que lleva incorporada el BASIC de tu ordenador. De hecho, incluso en este ejemplo, es realmente más rápido llamar a la función (o multiplicar el número varias veces por sí mismo) cuando la potencia es menor o igual que tres. Naturalmente, con DEF FN puedes abordar cálculos mucho más complejos que el de este ejemplo.

Aunque normalmente no interesará utilizar una fucnión para este tipo tan sencillo de cálculos, te puede servir para familiarizarte con el uso de DEF FN en tu ordenador, experimentando con versiones modificadas para definir nuevas funciones que dividan un número por 10, multipliquen por 2000 o calculen el seno. A continuación puedes utilizar las funciones matemáticas que tu ordenador ya tiene en su BASIC. Para comprobar si has definido una función correctamente, puedes compararla con lo que obtienes cuando el ordenador calcula la respuesta a un comando directo.

Durante los experimentos, es útil entender cómo funciona realmente DEF FN en tu ordenador.

NOMBRES DE FUNCIONES

Cada función que definas debe tener un nombre, utilizado para llamarla y emplearla en tus programas. El nombre está formado por una o dos letras situadas inmediatamente después de DEF FN. Por ejemplo: DEF FN ej (que empieza a definir una función llamada ej).

En realidad el nombre de la función en los **MSX** puede contener más de dos letras o números, pero sólo se tendrán en cuenta los dos primeros. Además el primer carácter de cada nombre debe ser una letra. Por ejemplo, se admite un nombre tal como ACIDO, pero el ordenador no apreciaría diferencia entre esta función y una que se llamara ACTOR, o cualquier otro nombre que empezase por AC.

DEFINICION DE PARAMETROS

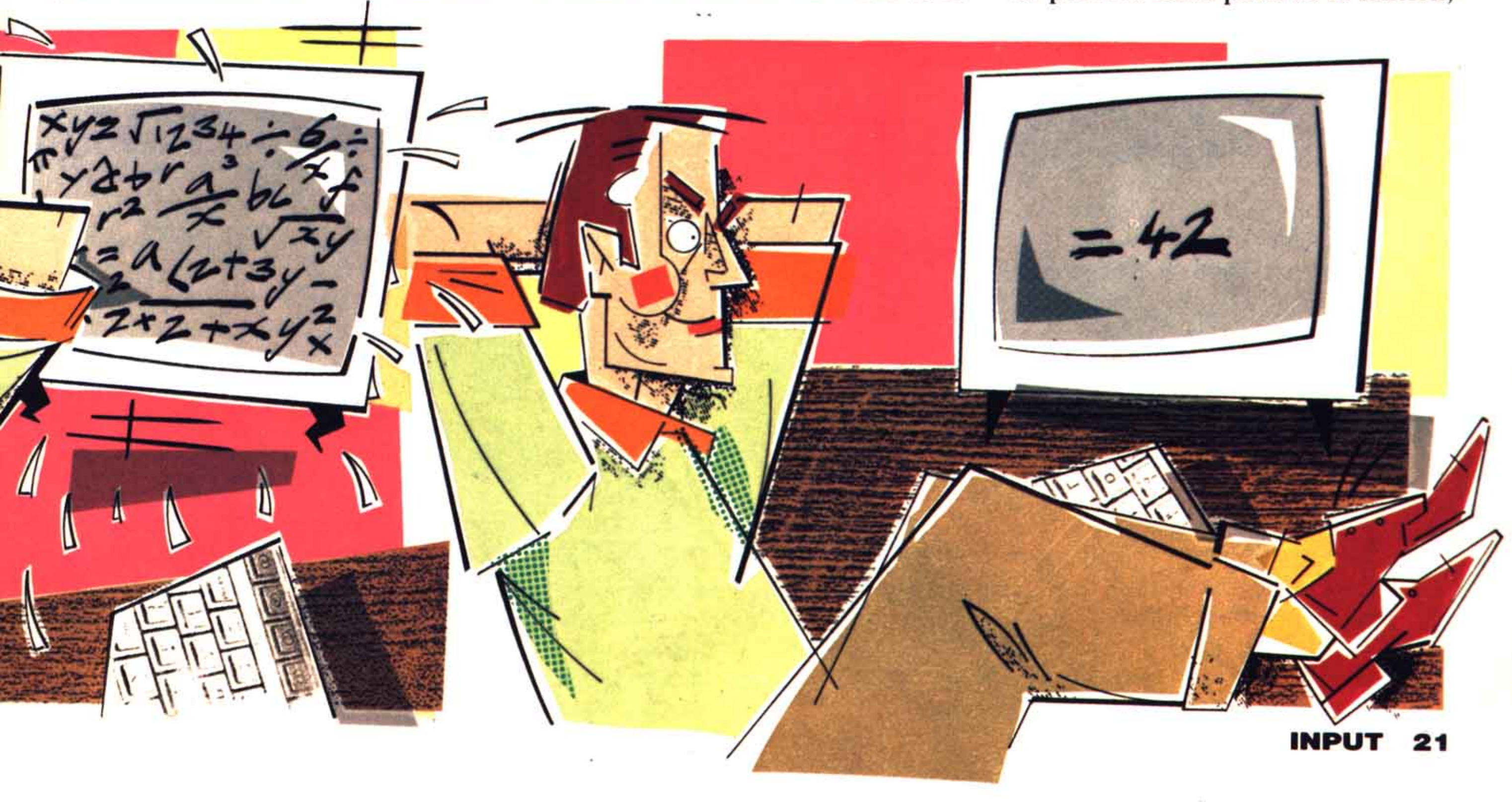
El siguiente paso para definir tu función es añadirle los «parámetros».Son los números o letras que figuran entre paréntesis después del nombre de la función.

Sirven para informar al ordenador de que la función utilizará esos números del programa en sus cálculos. En otras palabras, si empiezas definiendo tu función así:

DEF FN ej(a)

el ordenador esperará un número proporcionado por el programa. Asi, cuando hagas una llamada para utilizar la función en tu programa, deberán incluir un número o variable entre paréntesis (más adelante explicaremos cómo puedes usar realmente tus funciones definidas por usuario).

Tu **MSX** admite cualquier número de parámetros. Todos los demás valores numéricos que necesite la función han de ser valores numéricos reales que intervengan en la propia definición. Así, si quieres multiplicar un valor por 9.81 como parte de tu función,

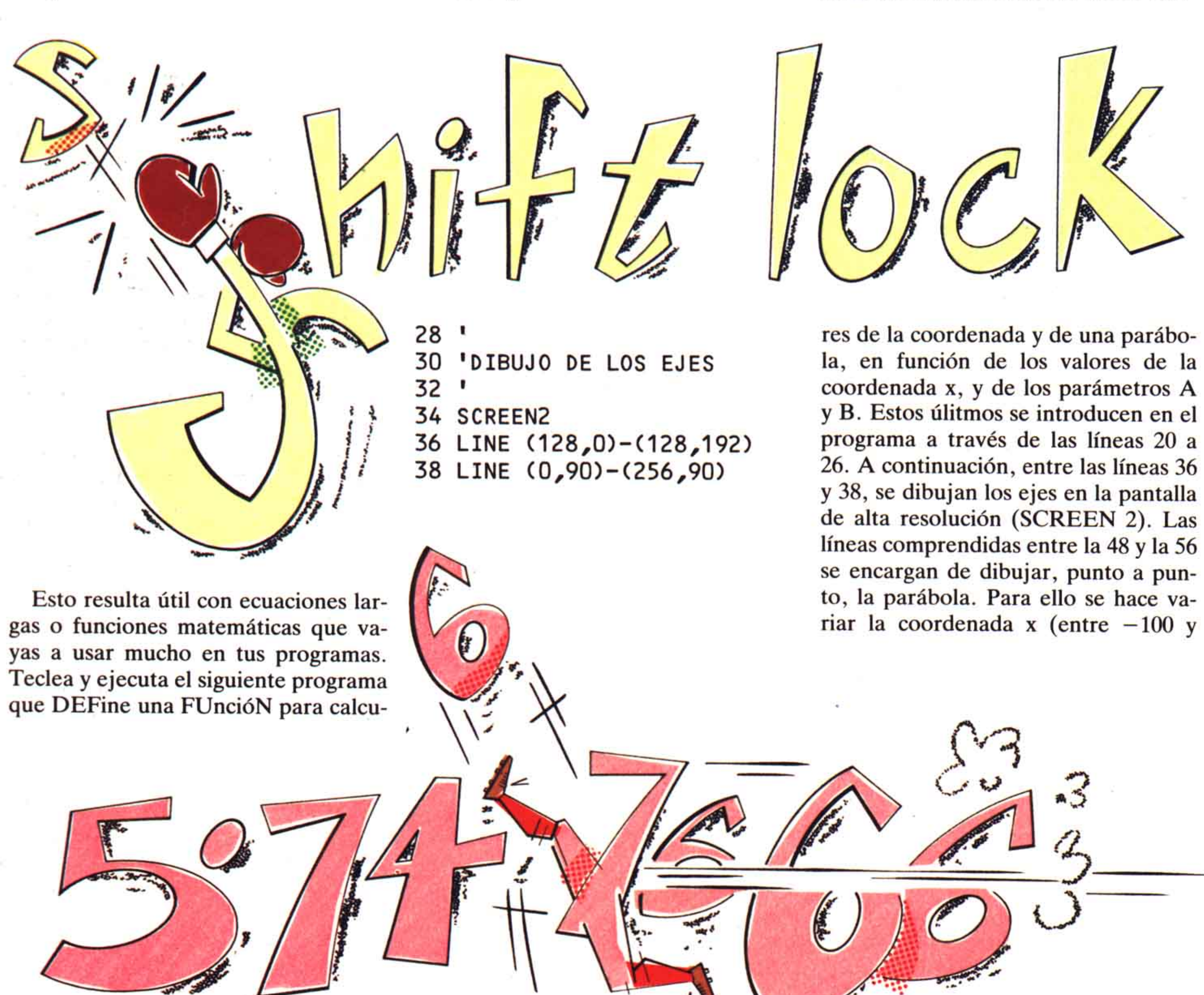


podrías utilizar un parámetro a para el valor que ha de ser cambiado por la función, y también usar 9.81 en la definición.

¿POR QUE USAR FUNCIONES?

¿Cuándo es mejor usar una función en una línea de programa en lugar de una instrucción directa?

La ventaja más clara de definir una función en vez de realizar una serie de cálculos, se presenta cuando tienes que realizar el mismo cálculo varias veces. Con una función definida por el usuario puedes ahorrar memoria y tiempo.

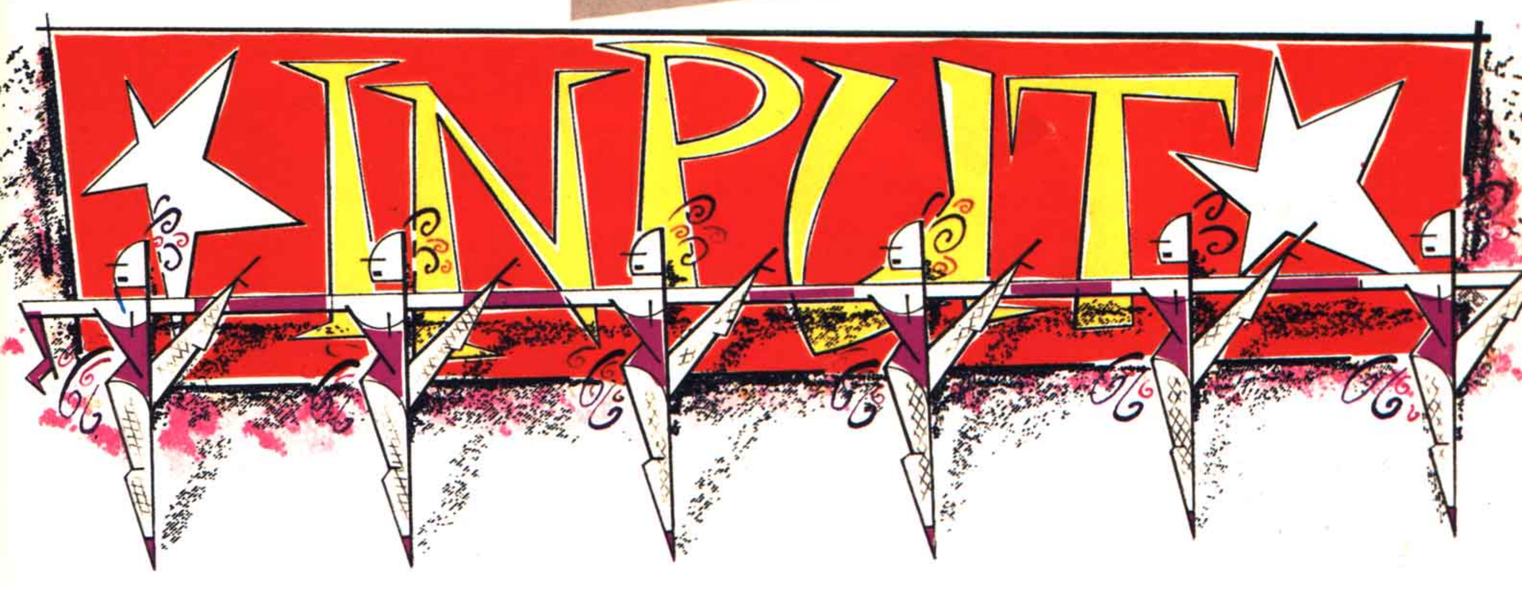


Esto resulta útil con ecuaciones largas o funciones matemáticas que vayas a usar mucho en tus programas. Teclea y ejecuta el siguiente programa que DEFine una FUnciónN para calcular valores a partir de la ecuación de una parábola.

Al ejecutarlo tienes que introducir en los INPUT unos cuantos valores, después de lo cual el programa utiliza la función para dibujar una parábola.

```
10 DEF FNY(X,A,B)=A*X*X+B
12 SCREEN0:CLS
14 '
16 'ENTRADA DE DATOS
18 '
20 PRINT"Valor de A"
22 INPUT"(entre 0 y 5)...";A
24 PRINT"Valor de B"
26 INPUT"(entre -5000 y 5000)
   ...";B
28 '
30 'DIBUJO DE LOS EJES
32 '
34 SCREEN2
36 LINE (128,0)-(128,192)
38 LINE (0,90)-(256,90)
40 '
42 'USO DE LA FUNCION
44 'Y DIBUJO DE LA PARABOLA
46 '
48 FOR J=-100 TO 100
50 CX=128+J
52 CY=90+FNY(J,A,B)/100
54 PSET (CX,CY)
56 NEXT
58 '
60 'ESPERA HASTA TECLA
   PULSADA
62 '
64 A$=INKEY$:IF A$="" THEN
   64 ELSE 12
```

El programa define la función en la línea 10. Esta función nos da los valores de la coordenada y de una parábola, en función de los valores de la coordenada x, y de los parámetros A y B. Estos últimos se introducen en el programa a través de las líneas 20 a 26. A continuación, entre las líneas 36 y 38, se dibujan los ejes en la pantalla de alta resolución (SCREEN 2). Las líneas comprendidas entre la 48 y la 56 se encargan de dibujar, punto a punto, la parábola. Para ello se hace variar la coordenada x (entre −100 y



100) y se calcula, mediante la función definida por el usuario, el valor correspondiente a la coordenada y. Por último, la línea 64 espera hasta que se pulsa una tecla antes de volver al principio del programa.

DALE MAS INTERES

Aquí tienes otro ejemplo de funcionamiento de una sentencia DEF FN, en forma de programa de utilidad que, aunque corto, puede ser de gran ayuda en el cálculo de los intereses que podrían producir tus ahorros.

El programa define una función que calcula la cantidad de dinero que tendrías después de transcurrir un tiempo determinado, y para un determinado tipo de interés. Tienes que introducir en los INPUTs la cantidad de dinero con la que empiezas, la tasa o tanto por ciento de interés y el período de tiempo.

```
10 DEF FNCALC(C,R,T)=INT
   (C*((1+R/100)^T)*100)
   /100
20 CLS:INPUT"Capital
   inicial...";CA
30 INPUT"Tasa de interes
   (%)...";RA
40 INPUT"No. de unidades de
   tiempo";TA
50 CLS:PRINT" Cap.";TAB(13)
   ;"Int.";TAB(23);"Tiempo"
60 PRINT CA;TAB(12);RA;TAB
   (22);TA
70 PRINT:PRINT"Cantidad total
   despues de intereses:"
   :PRINT FNCALC(CA,RA,TA)
80 LOCATE0,20:PRINT"pulsa
   una tecla"
90 A$=INKEY$:IF A$="" THEN
   90 ELSE GOTO 20
```

El funcionamiento de este programa es muy sencillo; define la función y pasa a las sentencias INPUT por medio de las cuales puedes introducir todos los detalles relativos a tus ahorros. Las variables utilizadas por el programa son T para el período de tiempo; C para el capital o cantidad de partida y R para el rédito o tipo de interés.

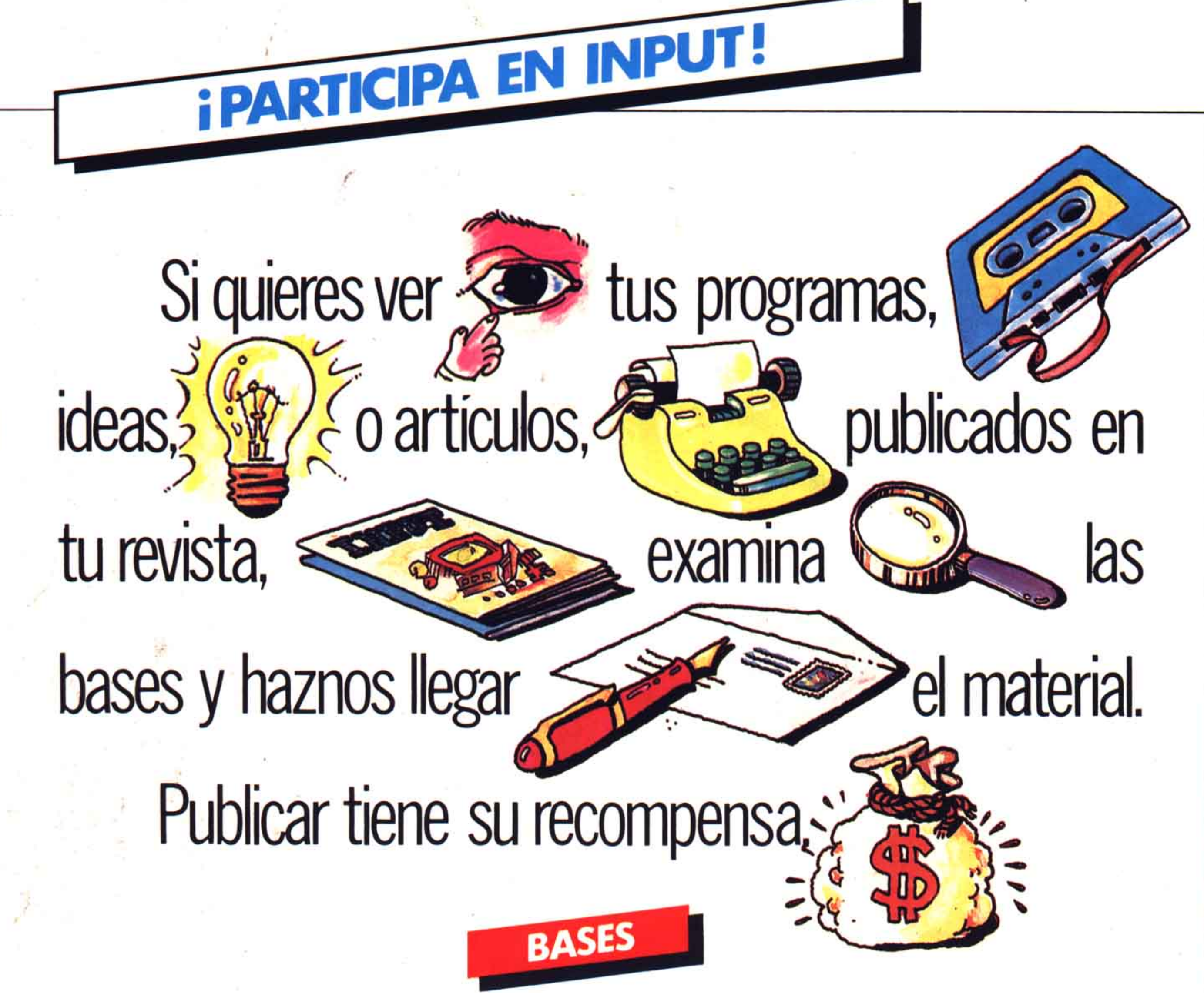
El valor calculado por el programa es el del interés compuesto, es decir, el interés producido en un período de tiempo se acumula al interés de los períodos anteriores. Esto es tenido en cuenta automáticamente por la función. Si, por ejemplo, la tasa de interés es de 10%, después de un período de tiempo tendrás la suma original más el 10% de la misma, o lo que es lo mismo, la suma original más 0,1 veces esta suma. Esta expresión aparece en la función como R/100+1. Así, en este caso el resultado después de transcurrir el primer período es 1.1 veces la cantidad con la que empezaste. Después de dos períodos el resultado será 1.1 veces la nueva suma de partida, que a su vez ya era 1.1 veces la suma original. Es decir, es 1,1*1,1 veces la cantidad con la que empezaste. Y 1,1*1,1 es 1,1 ↑ T. Esta es la segunda parte de la función, cuya forma completa, tal como aparece en el programa es (R/100+1) ↑ T. La parte restante de la expresión multiplica esto por la cantidad original C y redondea el resultado con dos cifras significativas, para lo cual multiplica por 100, utiliza la función INT y divide de nuevo por 100.

Puedes utilizar las dos fórmulas que siguen para crear dos funciones definidas por el usuario que no tiene tu ordenador. Son las fórmulas de ARC SIN y ARC COS:

```
ASN(X)=ATN(X/SQR(-X*X+1))
ACS(X)=-ATN(X/SQR(-X*X+1))
      +1.5708
```

También puedes usar usar fórmulas para otras funciones matemáticas.

¡PARTICIPA EN INPUT!



BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas; es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**
  en material microinformático a elegir por los afortunados.

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT MSX
Alberto Alcocer, 46, 4.° B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.


LAS BUENAS COMPAÑIAS
DE UN MSX
PROFESIONAL
APLICACIONES
ML-30 FD
CT-1501 E
ML-FX1/2
ML-10 DR
ML-10 MA
MSX System
Version 1.0
Copyright 1983 by Microsoft

ENSAMBLADORES DESENSAMBLADORES COMPILADORES

Ante todo aclaremos que en este artículo no se pretende enseñar a programar en código máquina o en ensamblador, ya que se requeriría de mucho más tiempo y espacio y, por otro lado, existen innumerables libros y publicaciones donde estos temas están ampliamente tratados. Se trata simplemente de introducir al lector en estos campos.

LENGUAJES DE PROGRAMACION

Cuando en las relaciones internacionales han de entenderse dos personas sólo caben dos soluciones: que uno aprenda el lenguaje del otro o que ambos hagan uso de un traductor.

En el mundo de los ordenadores ocurre exactamente lo mismo, aunque con algunos matices. Quede claro que bajo ninguna circunstancia la máquina va a aprender nuestro idioma. Somos nosotros los que debemos aprender un **lenguaje de programación** adecuado.

Podemos elegir el camino más directo, pero también el más penoso: **Lenguaje en código máquina**. Con él sacaremos el máximo provecho al ordenador, tanto en ahorro de memoria como en velocidad de proceso, pero al precio de tener que valernos exclusivamente de un lenguaje a base de **palabras** compuestas de ceros y unos.

La alternativa más cómoda es la de aprender un **lenguaje de alto nivel** como el BASIC, Logo, Pascal, Fortran, Cobol, etc. Nuestra forma de comunicarnos con el ordenador es más relajada y más humana ya que utilizamos




REM
LA COMPUTADORA.
©BY PEPE 86™


AHÍ TIENE SU NUEVO PUESTO DE TRABAJO, AMIGO. HALA, A ARRIMAR EL HOMBRO.
O.K
IUM


BIEN, HAGAMOS FUNCIONAR ESTE CACHARRO
SIN INSULTAR MACHO


CARAMBA, ES UNA COMPUTADORA DE ESAS QUE HABLAN. SI SE CREE MÁS LISTA QUE YO, SE EQUIVOCA

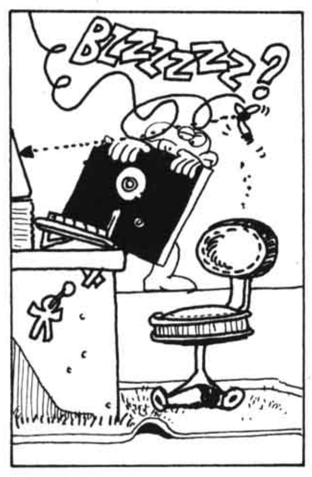

PERO... ¿Y SI NO SE EQUIVOCA?


...BUENO, PONGÁMONOS A TRABAJAR.
OYE ¿NUNCA TE DIJERON QUE ERES UN MALEDUCADO? NI SIQUIERA TE PRESENTAS Y EMPIEZAS A MANOSEARME.
10 REM
15 PRINT
20 PRINT "PEPITO COMPRÓ UN CERDITO"


ME LLAMO SIMPSON, SOY DE INGLATERRA, SIMP, PARA LOS AMIGOS. ¿DÓNDE PORRAS TIENES LA TECLA "ESC"?
ARRIBA, A LA DERECHA
GRACIAS
©by PEPE 86™


CARAMBA, ¿QUÉ ES ESTO?
¡NO!... NO TOQUES EL TECLADO! ES MI PRIMO XU12- DE LA OTAN, QUE ME LLAMA POR TELÉFONO
PRESS ANY KE

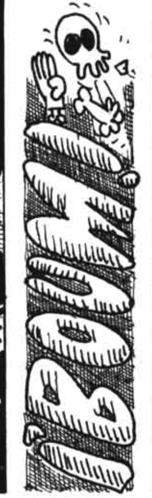

DISCÚLPAME, TENGO ALGO URGENTE QUE HACER


BZZZZZZ?


PRESS ANYKEY.
ESC
CTRL
RUN
4 5 6 7 8 9
R T Y U
O P


¿A VER QUÉ PASA?
A
TLAC


MIRE COMANDANTE, LA TIERRA, POR FIN.
MENOS MAL, A LA "GALÁCTICA" YA LE SALÍA BARBA
GALACTICA
MIREN, PARECE QUE ECHAN FUEGOS DE ARTIFICIO PARA RECIBIRNOS


BOOM!


PUES PARECE QUE LOS PIROTECNICOS, SE HAN PASADO CON LA PÓLVORA
END

un lenguaje estructurado de una manera análoga al inglés (todo el desarrollo del *software* se hizo inicialmente en E.E.U.U. y este país sigue controlando el mercado de ordenadores).

El inconveniente fundamental de los lenguajes de alto nivel es que requieren un paso intermedio de traducción para dialogar con el procesador. Todos necesitan ser **interpretados** o **compilados** y es aquí donde los programas pierden gran parte de su agilidad.

Existe un lenguaje intermedio, denominado **ensamblador**, bastante próximo al lenguaje en código máquina, pero con la gran ventaja de utilizar códigos mnemotécnicos, mucho más fáciles de manejar que los códigos binarios.

Al tratarse de un lenguaje orientado hacia la máquina (los de alto nivel se dice están orientados hacia el hombre) habrá tantos ensambladores como microprocesadores. Un programa escrito en ensamblador para el **Z80A** del **MSX**, **Amstrad** o **ZX-Spectrum** tendrá unas instrucciones ininteligibles para el **6502** del **Commodore**, por ejemplo.

PROGRAMANDO EN ENSAMBLADOR

Básicamente se trata de escribir un texto donde, renglón a renglón, y con ayuda de los mnemónicos correspondientes vayamos fijando los diversos pasos que queramos dar en nuestro programa. A este texto se le denomina programa en lenguaje ensamblador y en cada renglón hacemos figurar una sola instrucción.

El juego de instrucciones es demasiado extenso para incluirlo aquí, y puede conseguirse con facilidad en cualquier libro sobre código máquina del **MSX**. Cada instrucción (mnemónico propiamente dicho, más parámetros) tienen una codificación compuesta de un mínimo de 1 byte y un máximo de 4 bytes (un byte es una palabra de 8 bits ej. 11001010).

Un sencillo programa en ensamblador sería, por ejemplo el necesario para imprimir en pantalla un número determinado de veces un caracter específico.

```
ETIQUETA INSTRUCCION          COMENTARIO
-------- -----------          ----------
         LD   B,0     Situa el numero de repeticiones en
                      el registro B
BUCLE    LD   A,65    Situa el codigo ASCII del caracter
                      a representar (A) en el registro A
         CALL 162     Llamada a la rutina de impresion en
                      ROM
         DJNZ BUCLE   Decrementa 1 el registro B y salta
                      a BUCLE si el resultado es distinto
                      de cero
         RET          Regreso al BASIC
```

Figura 1

Planteamiento:

- Fijar el caracter a representar.
- Organizar un bucle del tamaño deseado.
- Representar el caracter en pantalla mediante la adecuada rutina de la ROM.

El programa en lenguaje ensamblador puede verse en la figura 1.

El programa anterior (parte izquierda), escrito en lenguaje ensamblador no es reconocible por el micro y simplemente nos ha servido para ayudarnos a plasmar nuestras ideas en forma parecida a como lo entenderá el ordenador. Ahora necesitamos **codificar** las instrucciones del programa. Esto es lo que hemos hecho en la figura 2.

Para facilidad del ejemplo se ha elegido un sólo byte como valor del bucle con lo cual sólo podremos repetir el caracter hasta 256 veces.

Se ha elegido el caracter «A» (cuyo código ASCII es 65), pero puede cambiarse por cualquier otro caracter siempre que su código ASCII esté comprendido entre 32 y 265.

Para codificar o **ensamblar** iremos a la misma tabla de donde hemos tomado los códigos mnemónicos y esta vez tomaremos los valores decimales correspondientes. Los valores de los operandos con sus posibles valores superiores a 255, como por ejemplo 23627, los convertiremos a hexadecimal y cada byte correspondiente lo expresaremos en decimal, cuidando de invertir el orden de los bytes (23627 decimal equivaldrá al número hexadecimal 5C 4B que introducido en la memoria sería 4B 5C hexadecimal o 75 92 decimal).

Una vez realizado el ensamblaje o codificación sólo resta elegir una posición de memoria, por encima de la permitida para código máquina, y colocar a partir de ella, y en posiciones sucesivas, los valores obtenidos. Podemos hacerlo directamente mediante POKE o utilizando cualquiera de los muchos **programas cargadores de código máquina** que existen publicados en libros y revistas. Todo depende del número de datos a manejar.

En este momento ya disponemos de un programa en código máquina, denominado **programa objeto**, corres-



```
ETIQUETA      INSTRUCCION      CODIFICACION
--------      -----------      ------------
              LD   B,0         6,0
BUCLE         LD   A,65        62,65
              CALL 00162       205,162,0
              DJNZ BUCLE       16,249
              RET              201
```

Figura 2

```
ETIQUETA   LENGUAJE ENSAMBLADOR   CODIGOS   COMENTARIO
--------   --------------------   -------   ----------
........   ....................   .......   ..........
```

Figura 3

pondiente al programa originalmente escrito en ensamblador y denominado **programa fuente**.

El programa cargador en nuestro caso sería:

```
10 CLEAR 200,49000
20 FOR N=0 TO 9:READ A:
   POKE 50000+N,A:NEXT N
30 DATA 6,0,62,65,205,162
   0,16,249,201
```

Una vez rodado el programa cargador podemos ejecutar la rutina en código máquina desde el BASIC. Incluimos un ejemplo donde a efectos de comparación podemos rodar el programa en código máquina que acabamos de generar y el equivalente en BASIC.

```
10 SCREEN 0:WIDTH 32
20 DEF USR0=50000
30 L=USR0(0)
40 PRINT:PRINT
50 FOR I=1 TO 256
   :PRINT "A";:NEXT I
60 END
```



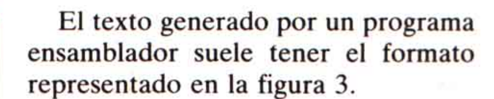
Cuando vamos a escribir un programa en ensamblador de cierta longirud nos encontramos con que el procedimiento descrito anteriormente para programar y ensamblar **a mano** resulta excesivamente laborioso y poco operativo.

En el lenguaje de código máquina (y ensamblador) no existen ni variables ni número de orden de instrucciones como en el BASIC. Un listado tampoco nos dice gran cosa, a simple vista.

Afortunadamente existen en el mercado potentes programas ensambladores que hacen por nosotros todo el trabajo duro. En próximos artículos hablaremos con detalle de uno de los mejores conjuntos ensambladores/desemsambladores (**MONS3/GENS3**).

Mediante estos programas podemos ayudarnos de variables, etiquetas (*labels*) (lugar donde direccionar una instrucción), procesos para edición, listado, documentación (posibilidad de meter texto auxiliar explicativo que ayude en la posterior reconsideración de un programa pero sin afectarle, en forma análoga a como lo haríamos con REM), etc.

También suelen incluir la posibilidad de ir probando el comportamiento de rutinas o aspectos parciales sin llegar a terminar el programa.



El texto generado por un programa ensamblador suele tener el formato representado en la figura 3.

DESEMSAMBLADO

Es justo la operación inversa de lo que hemos estado diciendo hasta ahora, es decir, generar un listado (texto) del programa ensamblador a partir de un programa objeto (código máquina).

Partiendo de un listado de los contenidos de posiciones sucesivas de memoria podemos, con ayuda de la tabla de codificación mencionada anteriormente retroceder a la situación original y recuperar el valor de los mnemónicos. Lo que nos será del todo imposible será recuperar el valor de las etiquetas y comentarios utilizados, por la sencilla razón de que se perdieron en el ensamblado.

Hay que poner especialísimo cuidado en comenzar por el inicio de una instrucción para no llegar a situaciones totalmente erróneas. Por ejemplo, de un listado en código máquina 209, 26, 213, 14, 47 obtendremos POP DE / LD A,(DE) / PUSH DE /LD C,47, pero si por error comenzásemos por la segunda cifra, es decir: 26, 213, 14, 47 llegaríamos a la falsa conclusión de que se trataba de las instrucciones LD A,(DE) / PUSH DE / LD C,47.

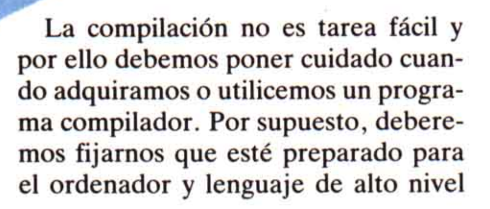
Existen igualmente programas comerciales desensambladores que nos facilitan enormemente la tarea y nos permiten a la vez ir documentando el programa para que una vez impreso sea más fácil su comprensión.

Si deseamos analizar un programa en código máquina el ideal es acudir al programa fuente. Por lo general sólo lo conseguiremos si somos nosotros los programadores o podemos entrar en contacto con ellos. En caso contrario sólo nos queda acudir al desemsamblado.

COMPILADORES

Trataremos de dar una idea clara sobre este tema en unas pocas palabras.

LD A, 4=3EØ4



Si lo que deseamos es no salirnos del BASIC porque nos resulta familiar y fácil de manejar, pero necesitamos ganar velocidad de proceso, como puede ser el caso de un programa que requiera mucho cálculo, o bien queremos acelerar la aparición en pantalla de los componentes de un juego que hemos diseñado y nos queda un poco lentorro podemos acudir a la **compilación** mediante uno de los diversos programas comerciales.

Compilar es pues, pasar directamente a un programa objeto desde otro descrito en un lenguaje de alto nivel.

La compilación no es tarea fácil y por ello debemos poner cuidado cuando adquiramos o utilicemos un programa compilador. Por supuesto, deberemos fijarnos que esté preparado para el ordenador y lenguaje de alto nivel que vayamos a utilizar, pero tampoco debemos pasar por alto las limitaciones de utilización.

Hay compiladores que sólo trabajan con números enteros y sólo reconocen un limitado grupo de instrucciones básicas. Otros son menos restrictivos pero más complicados, costosos y ocupan más memoria. En un gran número de casos los del primer tipo resultarán perfectamente válidos, quizá, en todo caso, con el retoque de alguna instrucción.

SPRITES PARA TUS JUEGOS

- COMO CONSTRUIR SPRITES
- MODOS DE PANTALLA
- DEFINICION DE UN SPRITE
- COMO DAR MOVIMIENTO A LOS SPRITES

Los *sprites* son gráficos que tu mismo puedes definir y que te ofrecen unas enormes posibilidades a la hora de programar todo tipo de juegos, desde los más sencillos a los más complicados. Te vamos a enseñar como construir y manejar *sprites* con tu MSX para que puedas incorporar a estos simpáticos «duendes» en cualquiera de tus programas.

Los *sprites* son objetos gráficos constituidos por una agrupación de *pixels* (puntos de pantalla). Estos puntos, que forman parte de un rejilla cuadrada, puedes seleccionarlos a voluntad a la hora de la definición del *sprite*.

El primer paso para definir el *sprite* consiste en dibujarlo sobre un papel cuadriculado. Previamente tendrás que decidir el tamaño que va a tener tu *sprite*, de los cuatro tamaños posibles que te ofrece tu **MSX**. Este tamaño se fija mediante la instrucción SCREEN, cuyo formato es:

```
SCREEN [modo de pantalla],
[tamaño de sprites]
```

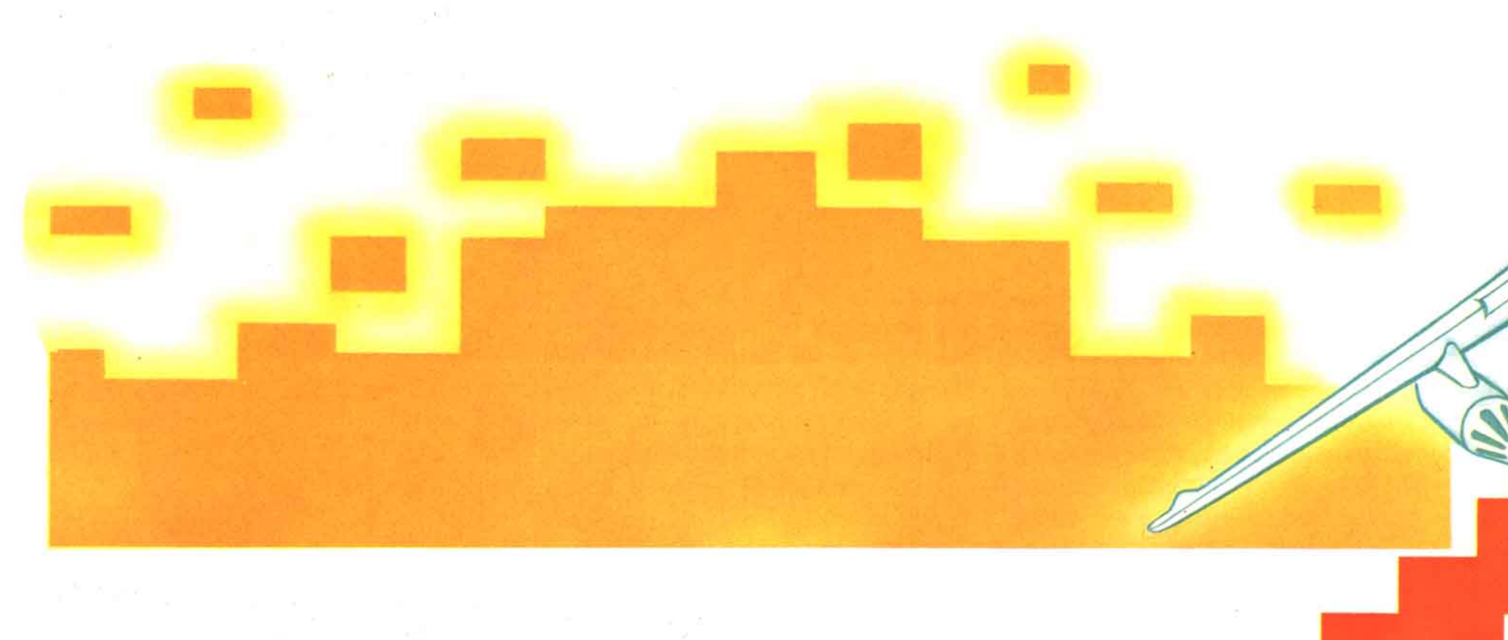
Hay 4 modos de pantalla en tu **MSX** que corresponden a los números 0, 1, 2 y 3. El modo 0 es un modo de texto en el que no puedes utilizar *sprites*. En cambio si puedes utilizarlos en los modos restantes.

Nosotros vamos a enseñarte el manejo de los *sprites* en el modo 2 que es el modo gráfico de alta resolución. En este modo puedes considerar la



pantalla como una rejilla formada por muchos puntos diminutos.

Hay 256 puntos horizontales por 192 puntos verticales, lo que nos pro-



porciona una pantalla con 49152 puntos en total.

Trabajando en este modo de pantalla (SCREEN 2) puedes elegir entre cuatro tamaños diferentes para tus *sprites*: 8×8, 8×8 (expandido), 16×16 y 16×16 (expandido). En el primer caso tu *sprite* podrá ser cualquier agrupación de 64 puntos (8 en horizontal × 8 en vertical), en el segundo caso tendrás el mismo número de puntos pero estos serán el doble de grandes. Si optas por el tamaño 16×16, tu *sprite* podrá ser cualquier agrupación de 256 puntos y si escoges este mismo tamaño expandido, seguirás teniendo una rejilla de 256 puntos para dibujar tu *sprite*, pero, al igual que antes, los puntos serán de tamaño doble. Cada uno de los tamaños de *sprites* se define mediante uno de los números 0, 1, 2 y 3. Así por ejemplo:

```
SCREEN 2,2
```

te coloca en el modo de pantalla 2 y con *sprites* de 16×16 puntos, mientras que

```
SCREEN 2,1
```

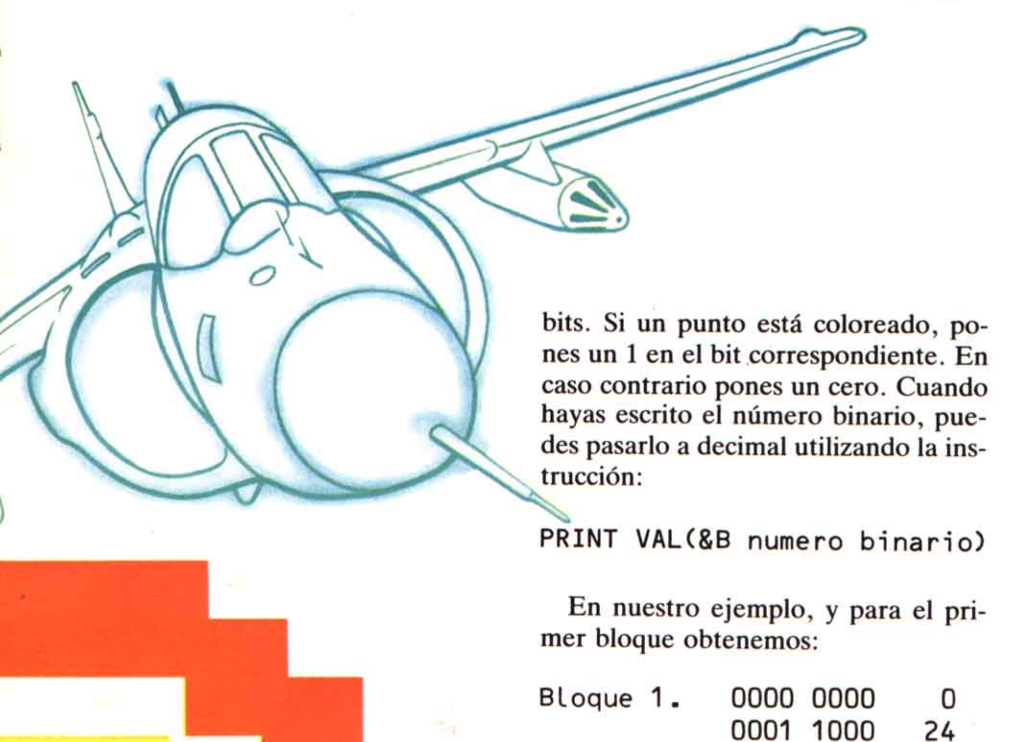
corresponde al modo de pantalla 2, pero los *sprites* en este caso serán de 8×8 puntos a doble tamaño, es decir, expandidos.

DEFINICION DEL SPRITE

Vamos a definir y a introducir en el ordenador un *sprite* de 16×16 puntos; para ello lo primero es coger un papel cuadriculado y delimitar una rejilla de 16×16 como el de la figura. Sobre esta rejilla dibujamos nuestro *sprite* teniendo en cuenta que los puntos que llenemos de color se verán, mientras que los que dejemos en blanco permanecerán invisibles. Vamos a dibujar por ejemplo un marciano, como el que puedes ver en la figura que se muestra en la página 24.

El siguiente paso consiste en convertir el dibujo en valores númericos para introducirlos en el ordenador. Para ello tienes que dividir el *sprite* en

4 bloques de 8×8 puntos e introducir los datos de cada bloque (1, 2, 3 y 4) unos a continuación de otros y en el orden señalado. Empezando por el bloque 1, tienes que coger cada fila y convertirla en un número binario de 8 bits. Si un punto está coloreado, pones un 1 en el bit correspondiente. En caso contrario pones un cero. Cuando hayas escrito el número binario, puedes pasarlo a decimal utilizando la instrucción:



```
PRINT VAL(&B numero binario)
```

En nuestro ejemplo, y para el primer bloque obtenemos:

```
Bloque 1.    0000 0000     0
             0001 1000    24
             0000 0100     4
             0000 0111     7
             0000 0101     5
             0000 0111     7
             0000 0111     7
             0000 0001     1
```

Haciendo lo mismo para los bloques 2, 3 y 4 obtenemos una serie de 32 valores decimales. Estos valores definen al *sprite*. Vamos a introducirlos en el ordenador mediante el siguiente programa.

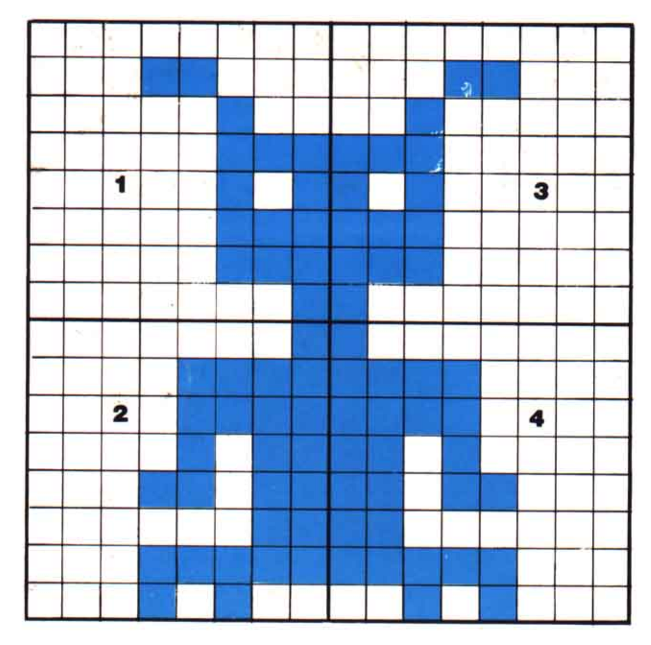
```
10 SCREEN2,2:CLS
15 FOR J=1 TO 32
20 READ A
25 S$=S$+CHR$(A)
30 NEXT J
35 SPRITE$(0)=S$
40 PUTSPRITE0,(100,100),15,0
45 GOTO45
1000 DATA 0,24,4,7,5,7,7,1
1010 DATA 1,15,15,11,27,1,31,
     26
1020 DATA 0,24,32,224,160,224
     ,224,128
1030 DATA 128,240,240,208,216
     ,192,248,40
```

Cuando ejecutes el programa (RUN) verás como al cabo de un momento aparece nuestro marciano en el centro de la pantalla.

El programa para el modo SCREEN 2 en la línea 10 define además el tamaño de 16×16 para el *sprite*. Entre las líneas 15 y 35 se leen los DATAs del *sprite* (líneas 1000 a 1020) y se asignan primero a la variable intermedia S$ y luego al *sprite* 0.

La línea 40 se encarga de colocar el *sprite* 0 en el plano 0, en las coordenadas x=100, y=100 y en color blanco (15).

Por último la línea 45 es un bucle infinito para evitar el salirnos del modo SCREEN 2.



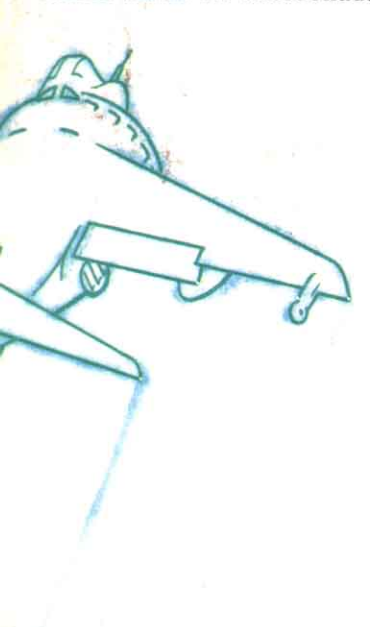
El marcianito de la figura se puede realizar mediante un sprite *de 16×16 puntos formado por 4 bloques de 64 puntos. Cada punto corresponde a un bit que tendrá el valor 1 donde hay dibujo y 0 donde no lo hay.*

MUEVE TUS SPRITES

La instrucción PUTSPRITE, sirve para colocar los *sprites* sobre la pantalla, pero también para moverlos. Cuando escribimos, por ejemplo:

PUTSPRITE 0,(x,y),15,0

estamos colocando en el plano 0 (hay 32 planos de *sprite*) y en las coordenadas (x,y) a nuestro *sprite* 0. Si hacemos variar las coordenadas x,y, tenemos que el *sprite* se mueve. Además, —y esta es una característica de los *sprites*— cuando lo colocamos en una nueva posición, dentro de un plano de *sprite*, no hace falta que borremos al *sprite* de su posición anterior, de ello se encarga el *chip* de vídeo, ahorrándonos trabajo y tiempo.



Añade las líneas siguientes al programa para ver cómo tu *sprite* se mueve aleatoriamente por la pantalla.

```
45 FOR J=1 TO 1000:NEXT J
47 XX=100:YY=100
50 R=RND(-TIME)
55 L=RND(1)*50
65 DX=1:IF RND(1)>.5 THEN DX=
   -1
70 DY=1:IF RND(1)>.5 THEN DY=
   -1
80 FOR J=1 TO L
82 IF(XX+DX)>255 OR(XX+DX)<0
   THEN DX=0
83 IF (YY+DY)>192 OR (YY+DY)<
   0 THEN DY=0
85 PUTSPRITE0,(XX+DX,YY+DY),
   15,0
86 XX=XX+DX:YY=YY+DY
90 NEXT J
100 GOTO55
```

El programa funciona de la siguiente forma: La línea 45 es un bucle de espera. En la línea 47 se almacenan en xx e yy las coordenadas de la posición actual del *sprite*. Estas coordenadas van a ir variando lo que va a producir el movimiento del *sprite*. En las líneas 50 y 55 se define una longitud aleatoria (L) para el camino que va a recorrer el *sprite* cada vez que se mueva. Las líneas 65 y 70 definen, también aleatoriamente, la dirección en la que se va a mover el *sprite*. Esta puede ser cualquiera de las 4 direcciones diagonales. Por ejemplo si DX=1 y DY=1, se incrementarán positivamente las coordenadas x e y, con lo que el *sprite* se moverá hacia abajo y a la derecha.

Si resultan DX=−1 y DY=1, el movimiento será hacia arriba y a la derecha.

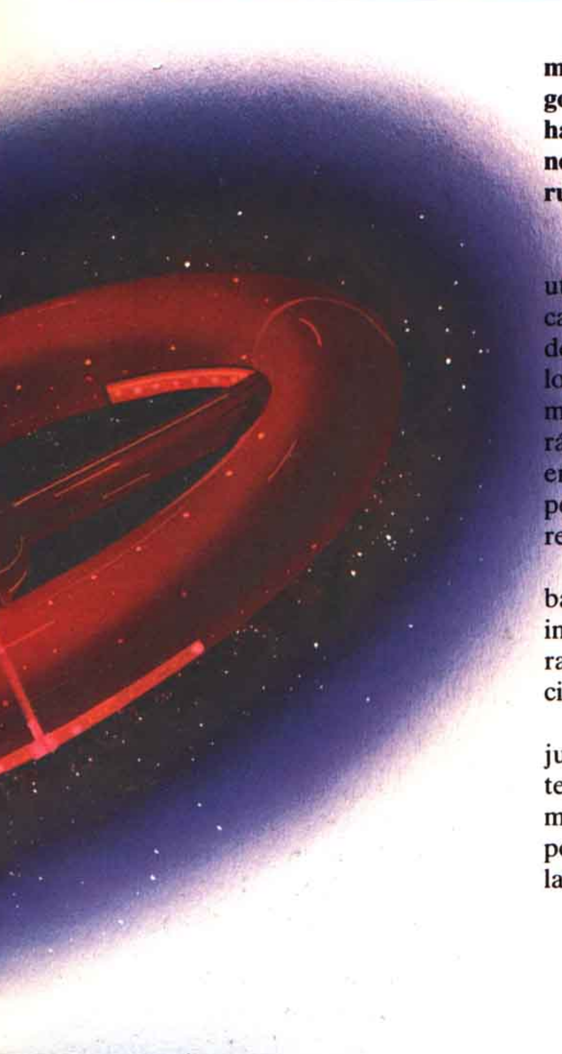
Entre las líneas 80 y 90 hemos incluido un bucle que produce el movimiento del *sprite*. Este bucle incrementa en uno las coordenadas xx e yy, tantas veces como indique la variable 1 y en la dirección que indiquen las variables DX y DY. Dentro del bucle, las líneas 82 y 83 definen unos límites para la pantalla, más allá de los cuales no puede moverse nuestro *sprite*.

Por último, al terminar el bucle, la línea 100 se encarga de volver al principio del mismo después de escoger nuevos valores aleatorios para L, DX y DY.

Como ves, mover tus *sprites* por la pantalla es algo muy sencillo. Sólo tienes que hacer uso de la instrucción PUTSPRITE.

ENEMIGOS MORTIFEROS Y EXTRATERRESTRES

- LAS RUTINAS PARA JUEGOS DE MARCIANITOS
- DIBUJO DE LOS ELEMENTOS
- INCORPORACION DE LOS ELEMENTOS



Desde Los Invasores, hasta los últimos juegos de marcianitos, los enemigos que atacan disparando siempre han sido un desafío. Aquí tienes la manera de crearlos e incorporarlos en una rutina compleja de juego.

Los juegos tendrán mejor aspecto si utilizamos algunas de las características de los gráficos de alta resolución de tu máquina, en lugar de servirte de los caracteres ordinarios. Los programas con gráficos de alta resolución serán más complicados que los que sólo emplean a los caracteres del teclado, pero ten por seguro que los resultados realmente merecen la pena.

Muchos juegos de marcianitos están basados en la presencia de enemigos invasores o extraterrestres que disparan contra tí, en vez de detenerse plácidamente a esperar que los aniquiles.

Seguidamente te presentamos un juego llamado Estación Espacial que te enseñará la manera de programar el movimiento aleatorio de un «invasor» por la pantalla, así como la forma de lanzar misiles contra un blanco.

El jugador dispone de 10 misiles con los que tendrá que destruir al marciano invasor. Este se mueve aleatoriamente por la pantalla descendiendo, desde la línea superior hacia la línea inferior, en la que se encuentra la nave del jugador.

Con los 10 misiles hay que destruir al marciano antes de que llegue a la línea de la nave. De no hacerlo, la nave del jugador quedará destruída y terminará el juego.

Tal como se presente aquí el juego, no está realmente completo, ya que le falta la puntuación y el cronometraje.

Pero esto se remedia fácilmente con los métodos que presentamos en los capítulos anteriores.

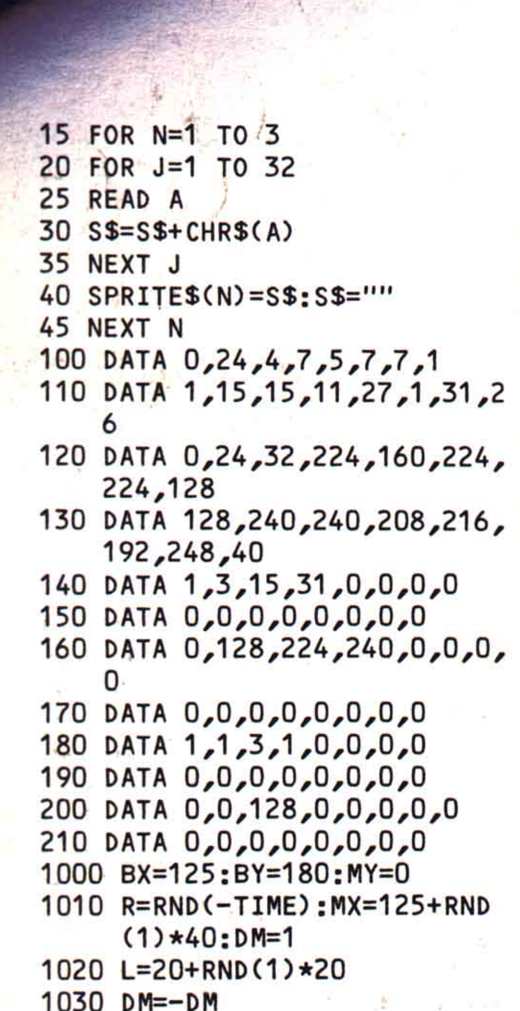
Por otro lado al estar escrito integramente en BASIC y sin cuidar demasiado de la velocidad, el juego puede resultar un poco lento. En cualquier caso es una primera versión en la que se ha pretendido aclarar conceptos sobre movimiento, dísparo de misiles, etc.

Para conseguir mayor velocidad recurriremos, en próximos capítulos, a

la utilización de rutinas de código máquina.

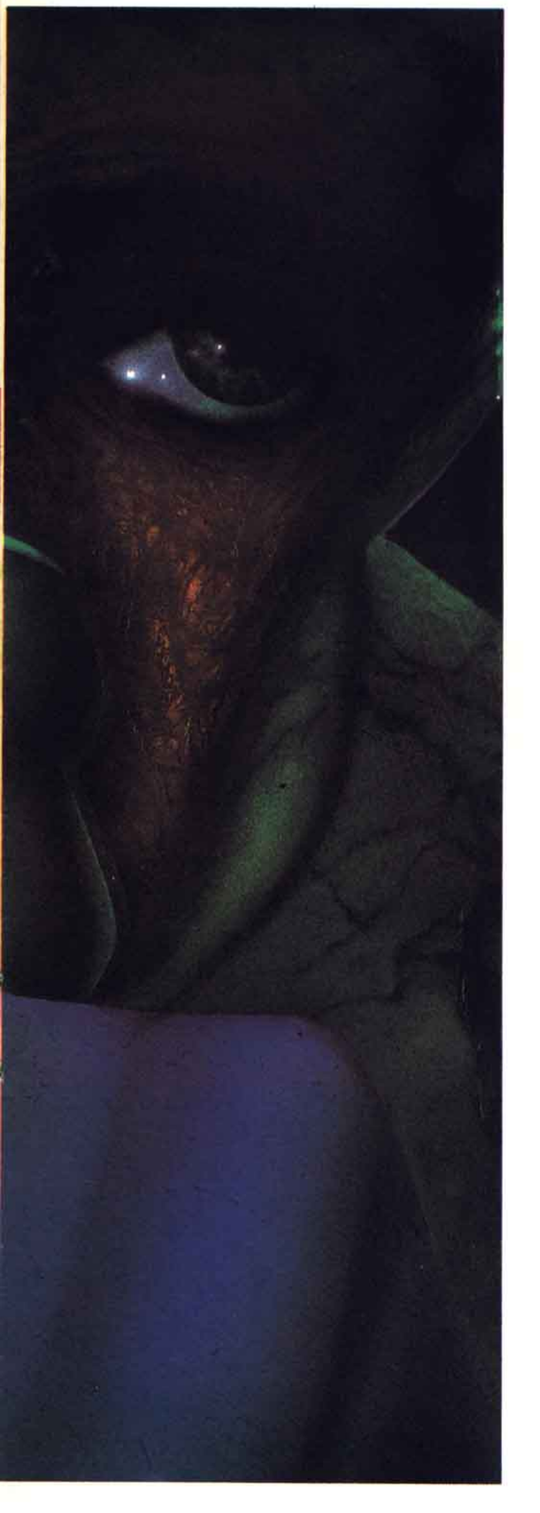
La presente versión del juego de la estación espacial utiliza los *sprites*, cuya información está contenida en un gran número de sentencias DATA cerca del comienzo del programa.

```
10 SCREEN2,2:CLS:SPRITEON:
   NM=5
```



```
15 FOR N=1 TO 3
20 FOR J=1 TO 32
25 READ A
30 S$=S$+CHR$(A)
35 NEXT J
40 SPRITE$(N)=S$:S$=""
45 NEXT N
100 DATA 0,24,4,7,5,7,7,1
110 DATA 1,15,15,11,27,1,31,2
    6
120 DATA 0,24,32,224,160,224,
    224,128
130 DATA 128,240,240,208,216,
    192,248,40
140 DATA 1,3,15,31,0,0,0,0
150 DATA 0,0,0,0,0,0,0,0
160 DATA 0,128,224,240,0,0,0,
    0
170 DATA 0,0,0,0,0,0,0,0
180 DATA 1,1,3,1,0,0,0,0
190 DATA 0,0,0,0,0,0,0,0
200 DATA 0,0,128,0,0,0,0,0
210 DATA 0,0,0,0,0,0,0,0
1000 BX=125:BY=180:MY=0
1010 R=RND(-TIME):MX=125+RND
     (1)*40:DM=1
1020 L=20+RND(1)*20
1030 DM=-DM
```

```
1040 MY=MY+10:IF MY>163 THEN
     GOTO2000
1050 FOR J=1 TO L
1060 MX=MX+DM*2
1070 A=STICK(0)
1080 IF A=3 THEN BX=BX+3:IF
     BX>200 THEN BX=200
1090 IF A=7 THEN BX=BX-3:IF
     BX<0 THEN BX=0
1100 A$=INKEY$:IF A$=" " AND
     MI<>1 THEN MI=1:SX=BX:SY
     =180
1110 IF MI=1 THEN SY=SY-10:IF
     SY<0 THEN MI=0:NM=NM-1:
     SY=180:PUTSPRITE3,
     (-20,SY),15,3
1120 IF NM=0 THEN GOSUB 2060
1130 PUTSPRITE1,(MX,MY),15,1
1140 PUTSPRITE2,(BX,BY),15,2
1150 IF MI=1 THEN PUTSPRITE3,
     (SX,SY),15,3
1160 ONSPRITE GOSUB 2010
1170 NEXT J
1180 GOTO1020
2000 SCREEN0:CLS:PRINT"El
     marciano ha terminado
     contigo":END
2010 SCREEN0:CLS:PRINT "HAS
     MATADO AL MARCIANO":
     PRINT
2020 PRINT"OTRA PARTIDA (S/N)
     ?";
2030 B$=INKEY$:IF B$="" THEN
     2030
2040 IF B$="s" THEN RUN
2050 IF B$="n" THEN END ELSE
     GOTO 2030
2060 SCREEN0:CLS:PRINT"Se
     acabaron los misiles":
     END
```



La primera línea del programa nos situa en el modo de alta resolución, limpia la pantalla, activa la detección de colisiones entre *sprites* y fija el número de misiles en 5 (NM=5).

Entre las líneas 15 y 45 el programa lleva a cabo la definición de los *sprites*. Para ello, y mediante dos bucles, se lleva a cabo la lectura de las características de los *sprites*, cuya forma viene dada en las sentencias DATA comprendidas entre la línea 100 y la 210.

Cada cuatro líneas DATA corresponden a un *sprite*. En total hay tres: uno para el marciano, uno para la estación espacial de lanzamiento de misiles y el último que corresponde al misil.

En las líneas 1000 y 1010, se establecen las coordenadas iniciales de la estación (BX,BY) y del marciano (MX,MY). La coordenada X del marciano (MX) se elige aleatoriamente en la línea 1010.

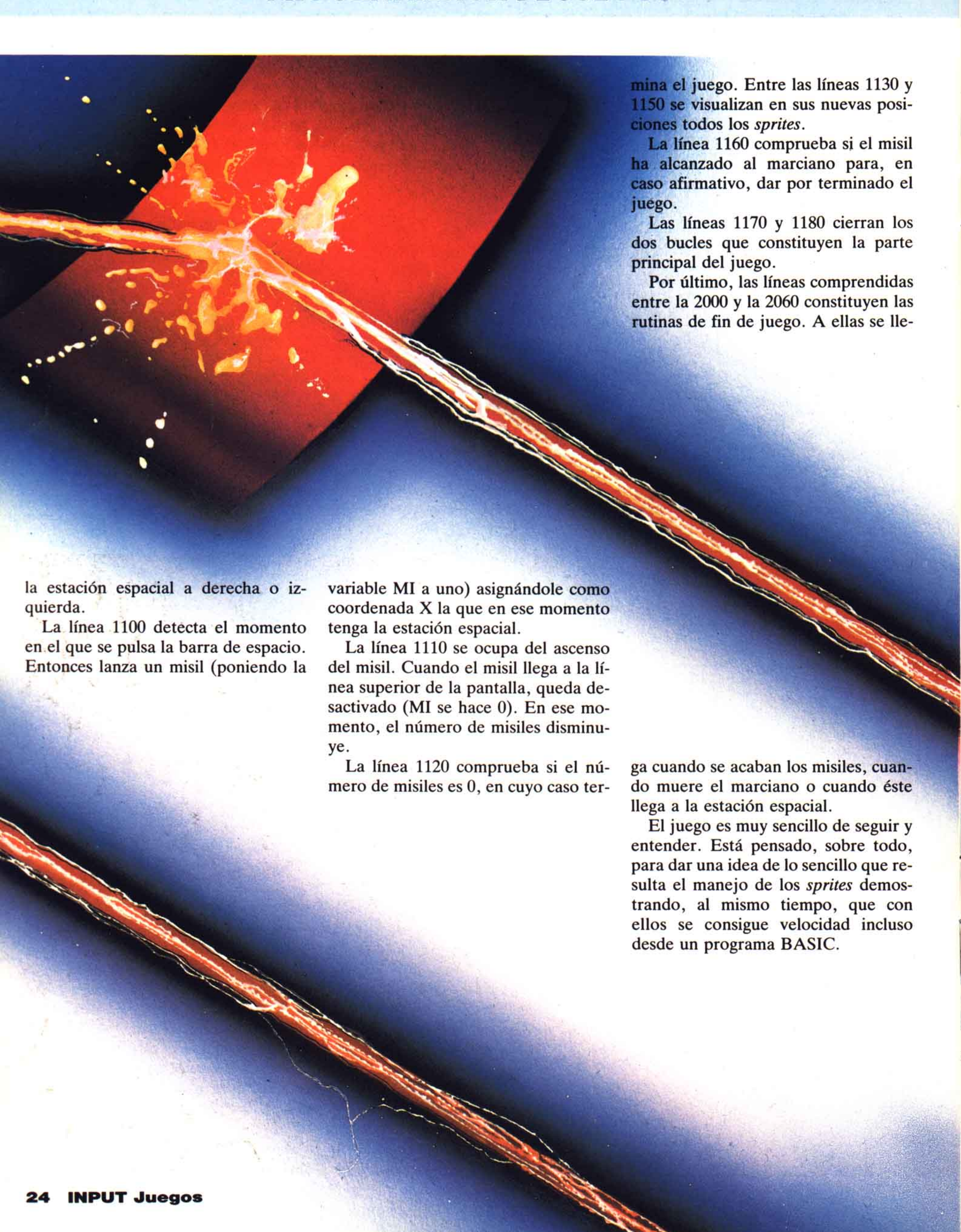
En las líneas 1020 y 1030 se elige aleatoriamente la longitud horizontal del desplazalmiento del marciano (L) y se alterna este desplazamiento de izquierda a derecha (al hacer DM=−DM). Es en la línea 1020 en la que se inicia el bucle principal del programa, dentro del que se mueven los *sprites*, se comprueba si ha habido colisiones, etc.



La línea 1040 hace descender al marciano al aumentar el valor de su coordenada Y (MY). Cuando esta vale más de 163, quiere decir que el marciano ha llegado a la estación espacial con lo que termina el juego.

Entre las líneas 1050 y 1070 hay otro bucle, que se encarga del desplazamiento horizontal del marciano al variar su coordenada X (MX) en la línea 1060.

Las líneas 1070 a 1090 se encargan de leer las teclas de cursor y desplazar



la estación espacial a derecha o izquierda.

La línea 1100 detecta el momento en el que se pulsa la barra de espacio. Entonces lanza un misil (poniendo la variable MI a uno) asignándole como coordenada X la que en ese momento tenga la estación espacial.

La línea 1110 se ocupa del ascenso del misil. Cuando el misil llega a la línea superior de la pantalla, queda desactivado (MI se hace 0). En ese momento, el número de misiles disminuye.

La línea 1120 comprueba si el número de misiles es 0, en cuyo caso termina el juego. Entre las líneas 1130 y 1150 se visualizan en sus nuevas posiciones todos los *sprites*.

La línea 1160 comprueba si el misil ha alcanzado al marciano para, en caso afirmativo, dar por terminado el juego.

Las líneas 1170 y 1180 cierran los dos bucles que constituyen la parte principal del juego.

Por último, las líneas comprendidas entre la 2000 y la 2060 constituyen las rutinas de fin de juego. A ellas se llega cuando se acaban los misiles, cuando muere el marciano o cuando éste llega a la estación espacial.

El juego es muy sencillo de seguir y entender. Está pensado, sobre todo, para dar una idea de lo sencillo que resulta el manejo de los *sprites* demostrando, al mismo tiempo, que con ellos se consigue velocidad incluso desde un programa BASIC.

ESTRUCTURA TUS PROGRAMAS

- ¿QUE ES LA PROGRAMACION ESTRUCTURADA?
- USO DE DIAGRAMAS DE FLUJO
- COMO USAR ESTRUCTURAS EN LOS PROGRAMAS BASIC

Un buen diseño hará que tus programas sean más fáciles de entender y que funcionen mejor. También puede marcar la diferencia entre que se ejecuten eficientemente o fallen estrepitosamente.

Cuando decides escribir tu primer programa para ordenador, lo normal es que sientas una incontrolable urgencia de sentarte ante el teclado y empezar a teclear inmediatamente alguna parte del programa.

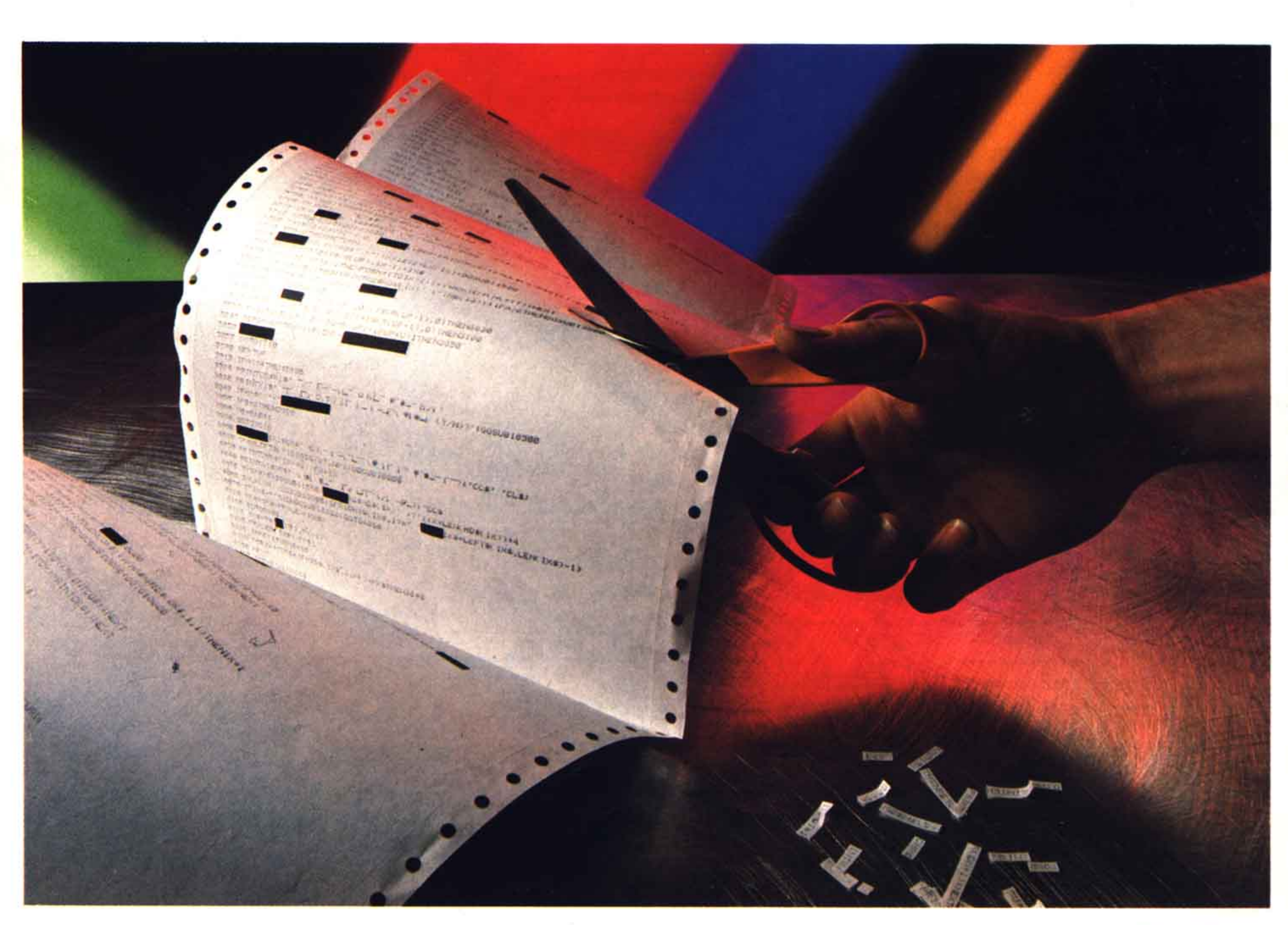
Puede ser que las primeras líneas que escribas funcionen bien. Te sentirás muy satisfecho contigo mismo y añadirás unas cuantas líneas más. Conseguirás que éstas funcionen también y seguirás añadiendo líneas aquí y allá, probando a medida que avanzas, hasta que el programa se extienda, probablemente a unos cientos de líneas.

Entonces, de repente, se produce el desastre. Añades unas líneas de programa y ya no funciona nada. No puedes encontrar ninguna razón por la que haya podido dejar de funcionar. Cambias esta línea, aquella otra, pero no hay nada que hacer.

Sin embargo, no te des por vencido. Hay maneras de hacer que esto no suceda, y lo único que hace falta es un poco de organización. Si sigues unas cuantas sencillas reglas cuando escribas tus programas, no tendrás el menor problema.

PROGRAMACION ESTRUCTURADA

El BASIC estándar fue diseñado para que fuera fácil de utilizar, pero tiene muy poca **estructura** comparado con otros lenguajes, lo cual hace más difícil escribir programas estructurados.



Las estructuras son los bloques básicos que utilizas para construir la forma general de tus programas. En el BASIC son IF ... THEN, FOR ... NEXT, GOTO y GOSUB. Ya sabes cómo se utiliza cada una de ellas por separado, pero nuestro objetivo actual es ponerlas juntas de una forma ordenada y legible.

Es muy fácil y rápido escribir unas cuantas líneas de programa que funcionen a la primera (o casi a la primera) y sean relativamente sencillas de entender por cualquiera. De hecho, si el programa es corto, no hace falta tomar especiales precauciones para estructurarlo. El problema se presenta cuando quieres escribir un programa largo que haga algo útil. En este caso, si empiezas escribiendo un fragmento de programa y le vas añadiendo más y más trozos, lo más probable es que termines irremediablemente perdido, a menos que tengas una memoria de elefante, naturalmente.

Lo que tienes que hacer es sentarte y diseñar el programa sistemáticamente. Casi todos los programas empiezan con una vaga idea de algo que tú quieres que haga el ordenador, y cuanto más complejo sea ese «algo» más vagas serán tus ideas. El concepto básico puede corresponder a un juego o un paquete de proceso de textos, o cualquier otra cosa tan amplia que te sea imposible mantener todas las ideas en la cabeza al mismo tiempo.

Para empezar sitúate en algún lugar lejos del ordenador, preferiblemente en otra habitación, para evitar tentaciones, y determina y escribe de una forma general lo que quieres hacer. Este es el principio de lo que se llama una **especificación de diseño**. Por ejemplo, podrías escribir:

Sistema de índices.

El programa permitirá crear, actualizar, borrar, ordenar y listar registros usando una palabra clave. Todo el conjunto de registros puede ser almacenado (SAVE) en un fichero y cargado (LOAD) desde el mismo.

A continuación tienes que dividir esta descripción tan general en unos cuantos pasos lógicos o **módulos**. Las operaciones que intervienen en cada módulo probablemente serán aún muy complicadas, por lo que también deberán ser fraccionadas en secciones más pequeñas, hasta que creas que los módulos de más bajo nivel son lo suficientemente cortos para ser manejados. La figura 1 muestra cómo podría hacerse esto para el Sistema de Indices.

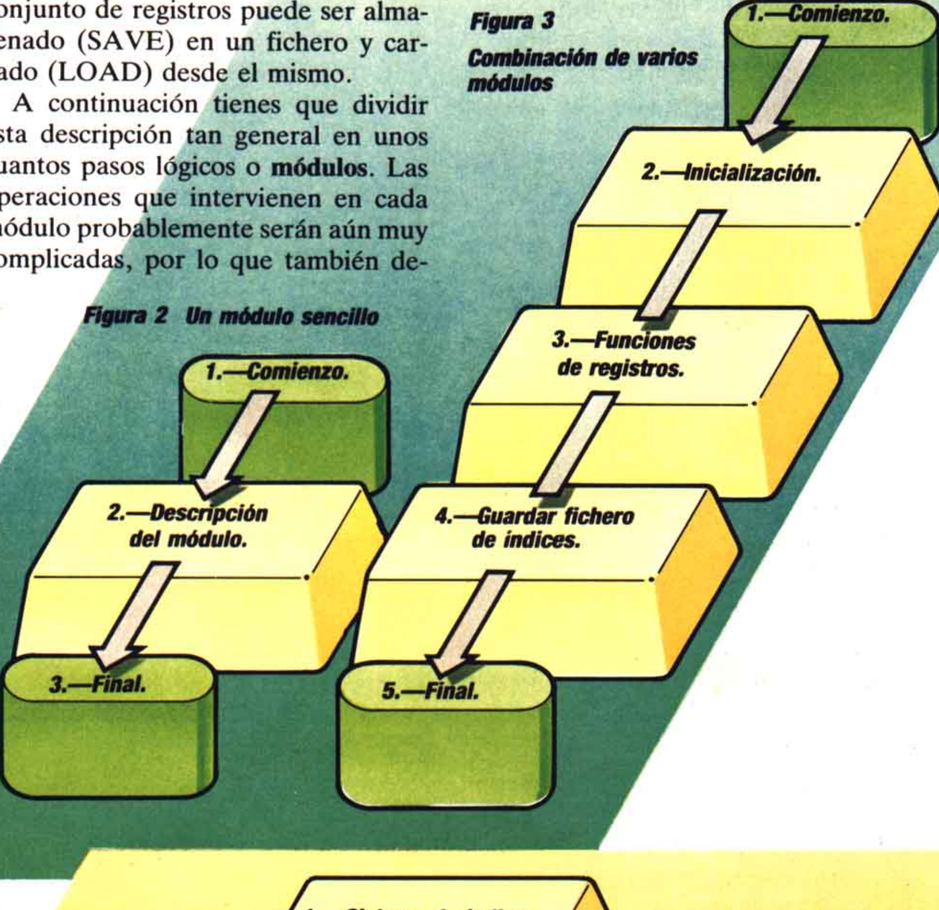


Figura 3 Combinación de varios módulos

Figura 2 Un módulo sencillo

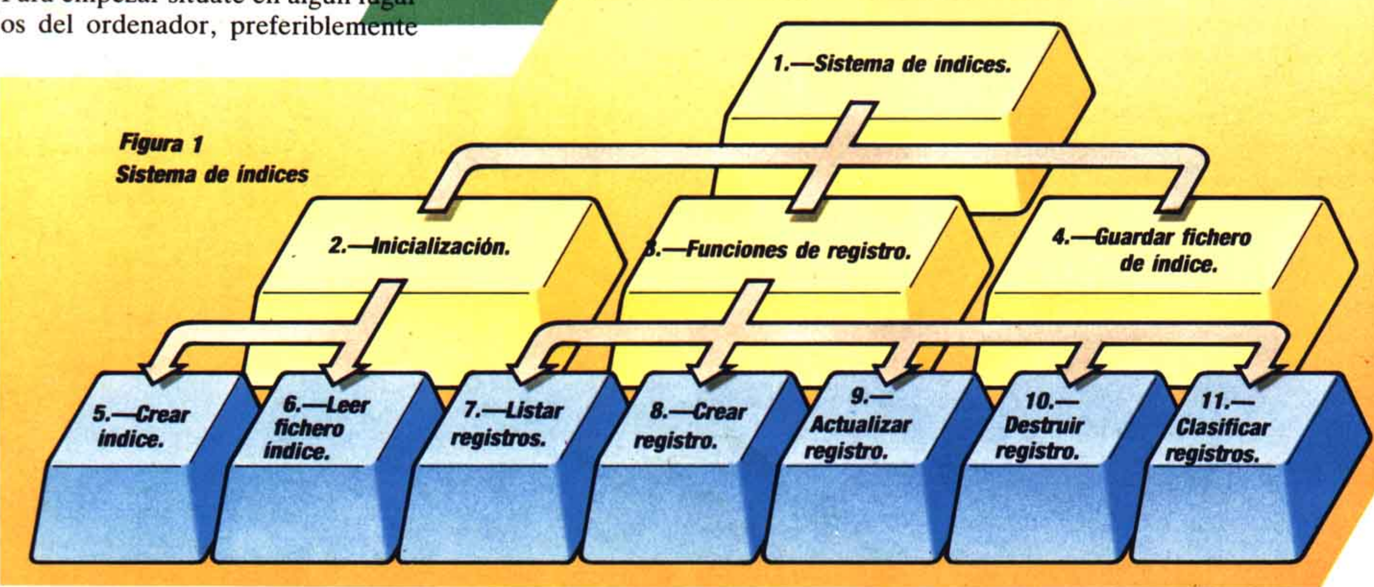


Figura 1 Sistema de índices

Cada una de las secciones más pequeñas no deberían superar la extensión de una página —unas 60 líneas— pero si la longitud es la mitad, mejor todavía. Cada uno de estos módulos de bajo nivel debe ser muy fácil de comprender. Eventualmente, cada módulo terminará como una subrutina de tu programa.

El proceso de fragmentar los problemas se irá haciendo más fácil con la experiencia. Y, como siempre, la mejor manera de aprender a hacerlo es intentarlo. Este método de dividir el problema se conoce como diseño de arriba a abajo. Empezaste por lo más alto (la descripción general del programa) y seguiste procediendo hasta lo más bajo (los módulos de más bajo nivel). Sin embargo, todavía no has decidido el orden de los módulos, es decir, el orden en que son ejecutados por el programa. Es lo que vamos a ver a continuación.

DIAGRAMAS DE FLUJO

Una manera de especificar el orden de los módulos consiste en utilizar **diagramas de flujo**. Son fáciles de usar y al ser esquemáticos, son fáciles de entender y seguir de una ojeada. Igual que el BASIC, no son estructurados en sí mismos, por lo que deben ser utilizados con precaución. Te puedes encontrar enredado y confundido con un diagrama de flujo, con la misma facilidad con que te ves atrapado en un enmarañado programa. Sin embargo, siguiendo unas cuantas reglas sencillas, los gráficos te pueden servir para ordenar un programa de una forma directa y estructurada.

La figura 2 muestra cómo puede ser especificado un módulo sencillo. El programa va siguiendo las líneas en dirección de las flechas y los bloques describen lo que sucede en cada etapa. Para ejecutar una serie de módulos en sucesión, no tienes más que añadir más bloques en el orden correcto entre Comienzo y Final (figura 3).

Un diagrama de flujo sencillo como éste vale para un programa en el que no se toman decisiones. No obstante, la potencia del ordenador se debe en gran medida a su capacidad de tomar decisiones dentro de un programa. Para esto es la familiar instrucción IF ... THEN que es una de las estructuras más usadas en BASIC.

1.—Inicio.
2.—¿Condición?
6.—Verdadera.
7.—Falsa.
3.—Sentencia 1.
4.—Sentencia 2.
5.—Fin.

Figura 4
Estructura IF ... THEN ... ELSE

Echemos un vistazo a todas las estructuras diferentes y a cómo pueden representarse con diagramas de flujo.

IF ... THEN ... ELSE

Aunque hay diferencias en la forma en que los ordenadores utilizan este comando, el conjunto IF ... THEN ... ELSE es la base de todas las decisiones que el ordenador tiene que tomar. Su diagrama de flujo se muestra en la figura 4 y en BASIC se codifica así:

```
100 IF condicion THEN
    sentencia 1  ELSE
    sentencia 2
```

El significado es: si la condición es verdadera, se ejecuta la sentencia 1, en caso contrario se ejecuta la sentencia 2.

Como puedes ver, el diagrama de flujo no tiene más que un punto de entrada y uno de salida. Esto facilita mucho la comprobación y la depuración de errores, pues conoces exactamente dónde comienza y acaba dicha sección del programa. Esta es de hecho una regla muy importante de la programación estructurada: para cada sección de programa debe haber sólo una entrada y una salida.

El BASIC de tu **MSX**, te ofrece la posibilidad de utilizar la construcción completa IF ... THEN ... ELSE. Prueba el siguiente ejemplo:

```
10 INPUT "Escribe un
   numero";A
20 IF A<5 THEN PRINT
```

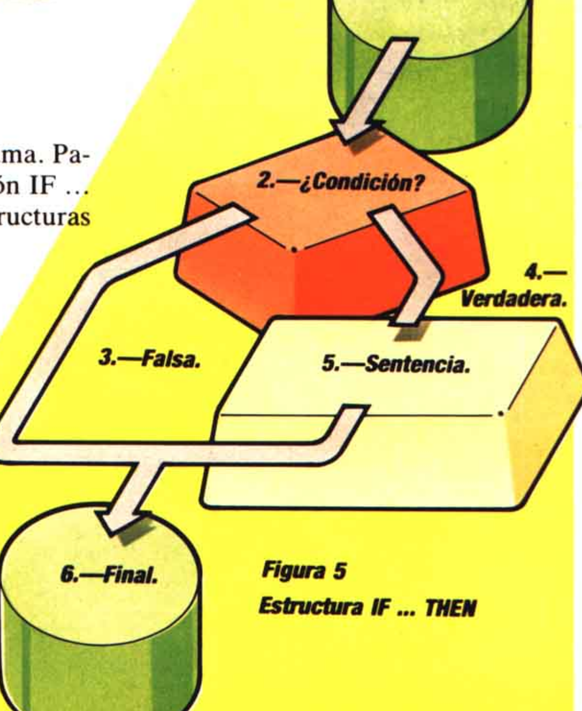


Figura 5
Estructura IF ... THEN

```
    "El numero es menor
    que cinco" ELSE PRINT
    "El numero es mayor o
    igual que cinco"
30 FOR J=1 TO 500:NEXT J
40 GOTO 10
```

Cada vez que introduzcas un número menor que cinco, en la sentencia 20 se ejecutará la parte correspondiente a THEN y se imprimirá la frase «El número es menor que cinco». Si el número introducido es mayor o igual que cinco, se ejecutará la parte correspondiente a ELSE y la frase será «El número es mayor o igual que cinco».

Sólo hay un pequeño problema y es que el conjunto IF ... THEN ... ELSE debe ocupar una sola línea de programa, es decir, no se puede, por ejemplo, incluir la parte IF...THEN en la línea 20 y dejar la parte ELSE para la línea 25. Todo debe ir en la misma línea.

Cuando lo que se quiere meter dentro de la construcción IF ... THEN ... ELSE ocupa más de una línea hay que recurrir a la siguiente construcción, que simula el comportamiento de ELSE mediante GOTO:

```
100 ...
110 IF condicion THEN GOTO
    140
120 sentencia 2:REM esta es
    la componente ELSE
130 GOTO 150
140 sentencia 1:REM esta es
    la componente THEN
150 ...
```

Naturalmente, los números de las líneas no tienen por qué ser los que hemos puesto aquí. Además en THEN y ELSE, se puede incluir más de una sentencia. Por ejemplo, aquí tienes una sección de programa para ordenar correctamente dos números. Constituye la base de una rutina de clasificación alfabética que veremos en otros artículos más adelante. En este caso, la componente ELSE contiene cuatro sentencias:

```
100 IF primero <= segundo
    THEN GOTO 160
110 temporal= primero
120 primero = segundo
130 segundo = temporal
140 orden$  = "erroneo"
150 GOTO 170
160 orden$  = "correcto"
170 ...
```

Sería equivalente escribir este ejemplo con ELSE, pero en tal caso habría que escribirlo todo en una línea. Se pueden utilizar sentencias múltiples siempre que estén separadas por dos puntos.

```
100 IF primero>segundo THEN
    temporal = primero:
    primero  = segundo:
    segundo  = temporal:
    orden$   ="erroneo"ELSE
    orden$   ="correcto"
```

Como ves, no resulta fácil leer o entender programas escritos usando sentencias largas. Por ello debes intentar evitarlas todo lo posible.

Por último, en muchos casos puede que no necesites para nada la componente ELSE. El correspondiente diagrama de flujo es el de la figura 5, cuya codificación es:

```
100 IF condicion THEN
    sentencia
```

que naturalmente es la sentencia IF ... THEN.

ESTRUCTURAS ANIDADAS

Las líneas con IF ... THEN ... ELSE pueden ir anidadas. Esto quiere decir que una o ambas sentencias entre las que puede escoger la sentencia con IF ... THEN, puede a su vez ser un IF ... THEN. Por ejemplo, la siguiente sección de programa lleva la cuenta de cuántos juegos van ganando dos jugadores y presenta en pantalla el resultado de cada juego:

```
100 IF T1<>T2 THEN GOTO 130
    ELSE PRINT "Empate":
    GOTO 190
130 IF T1<T2 THEN GOTO 170
    ELSE PRINT "Gana el
    primer jugador"
150 J1=J1+1
160 GOTO 190
170 PRINT "Gana el segundo
    jugador"
180 J2=J2+1
190 ...
```

Todas las estructuras pueden anidarse en cualquier combinación, y en teoría con cualquier profundidad. Sin

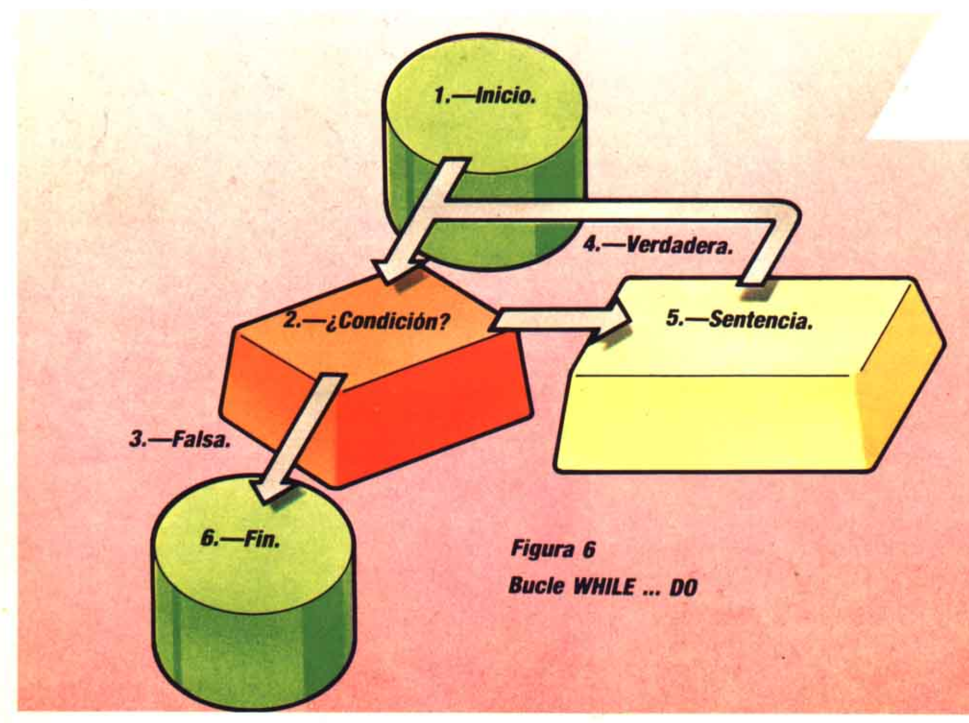


Figura 6
Bucle WHILE ... DO

embargo, cuanto más anidados pongas, menos legible será el programa; un límite práctico son tres a lo sumo cuatro estructuras anidadas. Si te encuentras con que te hacen falta más, divide el programa en módulos más pequeños o subrutinas.

Fíjate otra vez en el último programa. Es realmente difícil seguir lo que está haciendo, a pesar de que está perfectamente bien estructurado. Una forma eficaz de hacer más legibles las sentencias anidadas, es retranquear o escalonar (indentar) las líneas de programa. Esto es posible reescribiendo el último programa así:

```
100 IF T1<>T2 THEN GOTO 130
     ELSE PRINT "Empate":
     GOTO 190
130   IF T1<T2 THEN GOTO 170
       ELSE PRINT "Gana el
       primer jugador"
150    J1=J1+1
160    GOTO 190
170   PRINT "Gana el segundo
      jugador"
180   J2=J2+1
190 ...
```

Otra forma de hacer que la estructura de tu programa sea más clara es insertar líneas en blanco, separando las diferentes secciones y utilizar sentencias REM.

WHILE ... DO

La otra estructura esencial es la sentencia WHILE ... DO. Permite la existencia de bucles en un programa y es una de las mejores formas de crearlos. Hace que el ordenador haga (DO) algo una y otra vez mientras (WHILE) sea verdadera una cierta condición. No te preocupes si tu ordenador no tiene en su repertorio las palabras DO y WHILE. No están disponibles en casi ninguna formas de BASIC. Pero puedes construir una estructura que haga lo mismo, utilizando IF ... THEN y GOTO.

En la figura 6 se presenta el diagrama de flujo, y la codificación en BASIC es:

P y R

¿Tiene algún significado especial la forma de los bloques o cajas utilizados en los gráficos de flujo?

Sí, los diagramas de flujo se dibujan utilizando símbolos normalizados. Hay cinco formas principales:

1.º Los bloques redondeados se llaman bloques **terminales** y muestran dónde empieza y termina un programa.

2.º Los círculos son símbolos **conectores,** utilizados al principio y al final de un módulo.

3.º Los rectángulos son bloques de **instrucción** y contienen sentencias de programa.

4.º Los rombos son bloques de **decisión.** Siempre hay al menos dos caminos de salida de estos bloques, dependientes de una decisión que se toma dentro del mismo.

5.º Finalmente, los paralelogramos (no mostrados en este artículo) son bloques de **entrada/salida,** que indican cualquier información que entra al programa, y cualquier salida por pantalla o impresora.

```
100 ...
110 IF NOT(Condicion) THEN
    GOTO 140
120 Sentencia
130 GOTO 110
140 ...
```

Observa que la línea 110 figura IF NOT (condición)...; esto significa que se comprueba si la condición es falsa, al contrario de lo normal. Pero no hay problema. Si la condición es A = B, NOT(A=B) será entonces A<>B. Análogamente, NOT(A<B) será A> = B, etc. Realmente puedes escribir NOT (A=B) si te gusta y el ordenador entiende lo que quieres decir.

Aquí tienes como ejemplo un corto programa que utiliza un bucle WHILE y que puede servirte para contar el tiempo al hacer un huevo pasado por agua:

```
10 CLS:KEY OFF:COLOR 15,12
12 LOCATE 12,4:PRINT
   "TEMPORIZADOR"
14 LOCATE 4,7:INPUT"Cuantos
   minutos precisas ";T
16 LOCATE 5,19:PRINT"Pulsa
   ESPACIO para empezar"
18 A$=INKEY$:IF A$="" THEN
   18
20 CLS
22 PRINT TAB(12);
   "TEMPORIZANDO"
24 TIME=0
26 REM
28 REM COMIENZO DEL BUCLE
       WHILE
30 REM
32 IF INT(TIME/50)>=T*60
   THEN GOTO 54
34 LOCATE 7,10:PRINT
   INT(TIME/3000)
36 LOCATE9,10:PRINT":"
38 LOCATE10,10:PRINT
   INT(TIME/50) MOD 60
40 LOCATE 13,10:PRINT":"
42 LOCATE 14,10:PRINT
   ((TIME/5) MOD 10)
44 LOCATE 19,10:PRINT
   "Minutos"
46 GOTO 32
48 REM
50 REM FIN DEL BUCLE WHILE
52 REM
54 FOR N=1 TO 3
56 PLAY "o4v15cdcdcdcd"
58 NEXT N
60 END
```

El programa lleva la cuenta del tiempo transcurrido, leyendo la variable del reloj TIME. En la línea 32, al comienzo del bucle WHILE, se comprueba si el valor de TIME ha alcanzado el valor final que habíamos prefijado al principio del programa. Si no se ha alcanzado, se ejecuta la parte ELSE que se encarga de imprimir el tiempo en pantalla. Cuando se alcanza el tiempo prefijado, se ejecuta la parte THEN, que empieza en la línea 54, y suena el pitido de aviso. Es el momento de retirar el huevo del agua hirviendo.

CONCURSO INPUT/ANAYA MULTIMEDIA

¡DESCUBRE AL ASESINO!

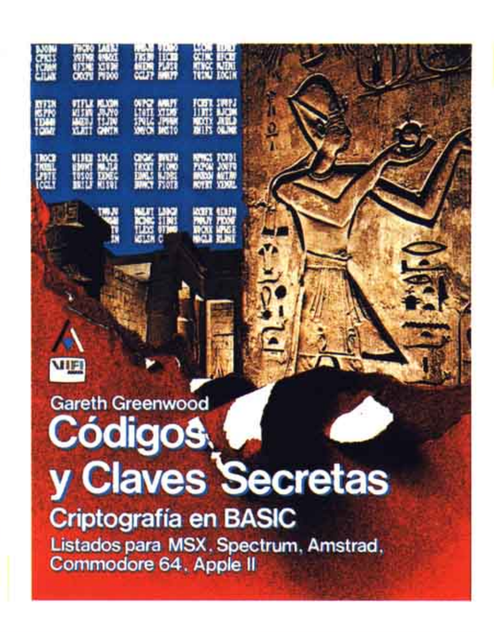
BASES:

Durante tres números sucesivos irán apareciendo en **INPUT** las tres partes del relato «El caso del anciano asesinado» junto con los mensajes cifrados que constituyen las respuestas parciales a la solución del crimen. Cada mes, quienes logréis descifrarlos participaréis en un sorteo consistente en 10 lotes de libros por un valor de 15.000 ptas., a elegir del fondo editorial de **Anaya Multimedia**, y una suscripción gratuita por un año a **INPUT**. Habrá tres sorteos, de tal forma que no será necesario haber descifrado el enigma del mes anterior para optar al premio.

Un fabuloso premio será la guinda final de este pastel. El descubridor de las motivaciones del crimen visitará al próximo **PCW Show** a celebrar en Londres. En caso de haber más de un acertante, recurriremos a la inevitable fórmula de sorteo.

La solución del primer mensaje cifrado deberéis enviarla, junto con vuestros datos personales, a la Redacción de **INPUT** antes del 15 de junio.

Las decisiones del jurado serán inapelables, dándose en las páginas de **INPUT** cumplida cuenta de la marcha del concurso.



*Nota: El libro de reciente aparición «Códigos y claves secretas», de **Anaya,** contiene todos los programas útiles para esta labor. Aunque si has trabajado para los servicios de inteligencia de alguna potencia no te será de gran utilidad.*



EL CASO DEL ANCIANO ASESINADO

Repasé la casa y comprobé que el viejo no fumaba, aunque no carecía de algunos vicios cuya enumeración no aportaría nada a este informe.

—El ladrón o los ladrones debieron dejarse un cigarro encendido —comentó mi ayudante.

—Los ladrones no suelen fumar mientras trabajan —respondí.

Di un par de vueltas más, que sólo me sirvieron para comprobar la porquería que es capaz de acumular en una casa un hombre solitario, aun cuando sea tan rico como aparentaba nuestro cadáver maniatado. Revisé de nuevo el interior de la caja fuerte. Había cartas de amor, recortes de periódicos, facturas sin saldar, apremios de pago, títulos de propiedad, una dentadura postiza y una póliza de seguro. Me guardé la póliza y dejé lo demás donde estaba.

En esto, llegó el juez con intención de levantar el cadáver. Tuvo que esperar un poco, porque nuestros expertos no habían terminado de estudiar el modo en el que el viejo había sido maniatado. Al poco, apareció también por el piso un anciano muy parecido al muerto y que dijo ser hermano suyo.

—¿Cómo se ha enterado Vd. de todo esto? —pregunté.

—Vivo en el piso de arriba. Estaba durmiendo la siesta, pero me ha avisado un vecino —respondió llorando frente al espectáculo.

Entonces, mi ayudante me pasó una nota que decía:

```
XNLZNJSYJ HNKWFIT HTQZRSFX HTRUQJYFX XFI
KQ UISIN MFTSN QXZSJ FSFWS TFRSJ JXJYS Z
XFJV FISNF XJJIW JNTTS IXUTF NJRQW YQZTW
 SNTYN JQ
```

Desde la comisaría telefoneé a la casa de seguros de la póliza y me confirmaron lo que ya había leído: que cubría un seguro por robo de una serie de joyas valoradas en quince millones de pesetas, joyas que, según la información de la casa de seguros, permanecían en una caja fuerte de la casa del anciano muerto. Al día siguiente visité a su hermano y le sometí a un complicado interrogatorio, complicado porque el pobre viejo no paraba de llorar.

Juan José Millas

EL RATON, UN SIMPATICO ACCESORIO

El reto aparece casi al mismo tiempo en que comienza a desarrollarse la ciencia informática. La comunicación hombre/ordenador siempre ha sido de naturaleza artificiosa. Desde luego, un teclado no es la forma más amigable, aunque pueda ser rápida, de acceder a un ordenador.

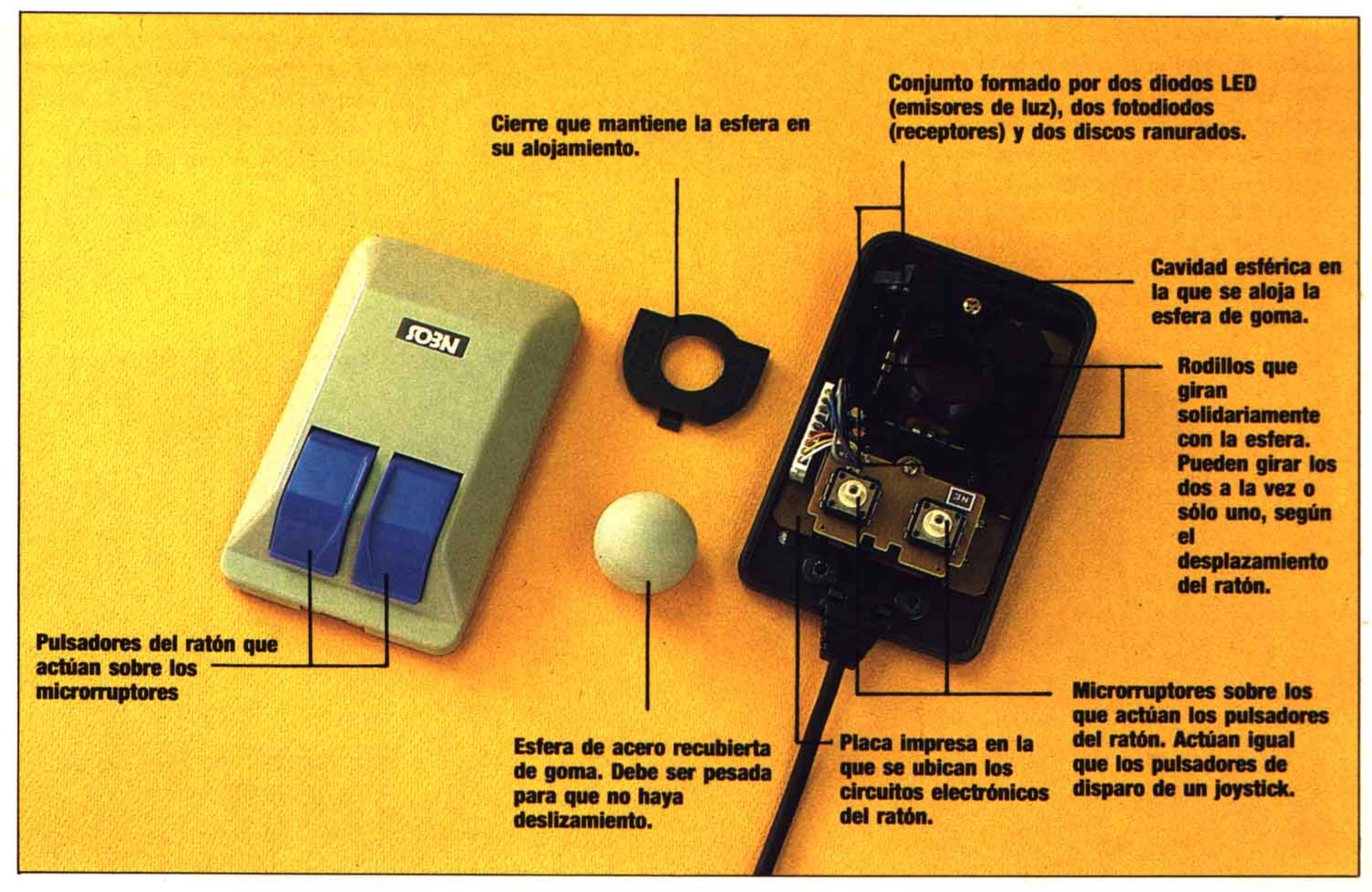
Sin embargo un teclado representa un enorme avance con respecto a las tarjetas y cintas perforadas que se empleaban con el mismo fin. La interactividad estaba vetada a prácticamente cualqueir usuario. La divulgación de los lenguajes de alto nivel interpretados (por contra a compilados) supone un enorme paso adelante en las comunicaciones hombre/ordenador. Se puede escribir una sentencia y conocer rápidamente si es comprendida y cómo por el sistema.

Los programas diseñados para cubrir una aplicación específica se ponen al alcance de cualquiera que precise los servicios que puede ofrecer un ordenador, sin necesidad de conocer una palabra sobre programación. En este gran apartado se pueden incluir también los videojuegos en cinta o *diskette*. Es particularmente en esta categoría donde el teclado se muestra antipático, sobre todo en los juegos de acción (en los que más cuentan los reflejos) que precisan mayor interactividad capaz de aceptar respuestas inmediatas. Hasta ahora, el periférico más divulgado que cuenta con las características de facilidad de manejo y «amigabilidad» con el usuario es el *joystick*.

No obstante, para determinados juegos y aplicaciones el *joystick* sigue presentando carencias y a veces resulta incluso incómodo. Cabe resaltar que una gran mayoría de modelos nos obligan a utilizar ambas manos para su control «una ha de agarrarlo con fuerza por la base».

Es en plena meca de la microelectrónica e informática, en California, donde el **Instituto de Investigación de Stanford** (SRI) comenzó a desarrollar un nuevo dispositivo más identificable con el usuario, a finales de los sesenta. Una de las soluciones consistió en una diminuta cajita capaz de ajustarse en la concavidad formada por la palma de la mano. En la parte anterior asomaban dos pulsadores, semejando a un par de ojos, también disponía de un cable de conexión al ordenador que recordaba a un rabo. Esta unión de apariencias es la que llevó a bautizar al dispositivo como «ratón».

Hasta hace bien poco el ratón no



era un accesorio demasiado utilizado con los microordenadores. Fué el **Lisa** de **Apple Computer** quien dió este primer paso con intención de popularizarlo y posteriormente lo hizo compañero imprescindible del **McKintosh**, una versión de menor precio del anterior. En los últimos tiempos se ha venido utilizando cada vez más, pero es realmente ahora cuando comienza a llegarle la oportunidad a ordenadores de precios más modestos.

En esencia, un ratón se comporta como un *joystick* por lo que al ordenador afecta. Inicialmente fué concebido para ser un dispositivo que se desplaza por una superficie plana —una mesa— en cualquiera de los sentidos y direcciones, bajo el control directo de una mano. Mediante *software* se conseguiría que el cursor se desplace por toda la pantalla siguiendo el mismo movimiento aplicado del ratón. Esto es obviamente mucho más rápido que el empleo de las teclas de cursor para hacer lo mismo.

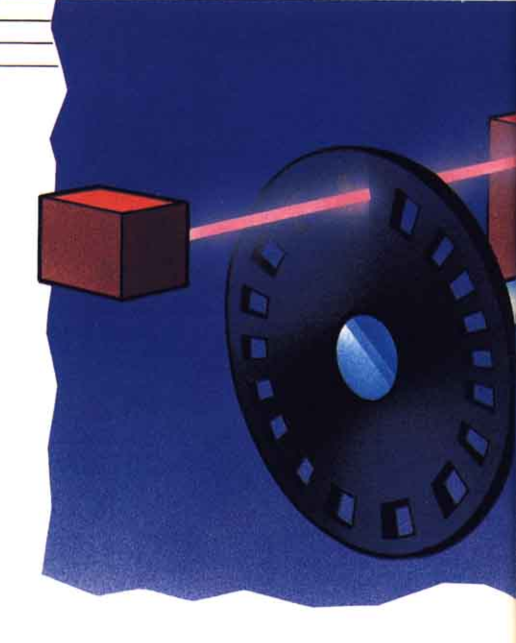
Los dos pulsadores, que se encuentran siempre al alcance de algún dedo, actúan de modo similar a los de «fuego» que incorporan los *joysticks*.

Un funcionamiento de este tipo permite que un usuario no experimentado pueda sacar rendimiento inmediato a programas de aplicación manejables a partir de menús. El cursor tomaría la apariencia de una flecha, por ejemplo, en lugar del clásico cuadrado. Así, moviendo la flecha hasta señalar la opción elegida, no hay más que presionar el botón de disparo y continuar eligiendo opciones, introduciendo datos, etc.

Se podría argumentar que esto también se puede conseguir con el *joystick* y en efecto es así, pero a la larga resulta más incómodo y menos «serio» cuando no se trata de programas de juego. Incluso con dicho tipo de programas se ejerce más cómodamente el control, porque los movimientos intuitivos se reflejan de modo más inmediato.

COMO FUNCIONA EL RATON

El funcionamiento del ratón responde a principios bastante sencillos. La base de todo la constituye una esfera maciza de acero, recubierta de goma para que su adherencia sea máxima, que puede girar libremente en el interior del ratón. Esta esfera asoma ligeramente por una abertura circular, en la base del ratón, de forma que al desplazar éste sobre una superficie plana, como la de una mesa, la esfera gira en el interior de la caja. Al girar, la esfera comunica su movimiento a un par de cilindros situados perpendicularmente entre sí, en el interior del ratón. Estos cilindros, llevan acoplado a su eje un disco ranurado que gira al mismo tiempo que lo hacen los cilindros.

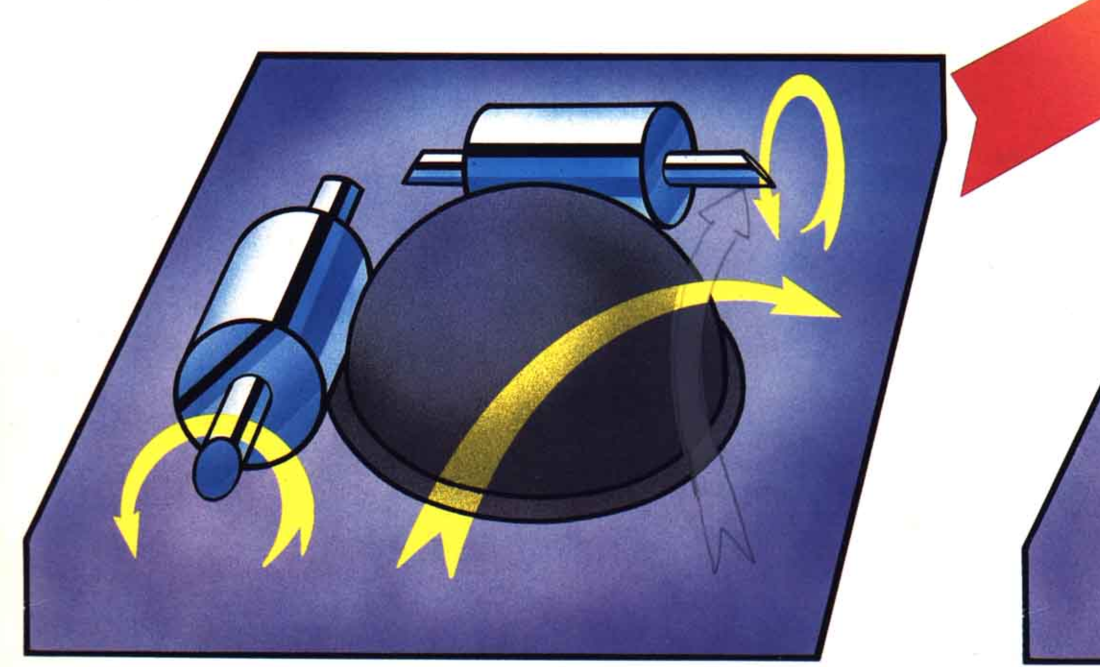


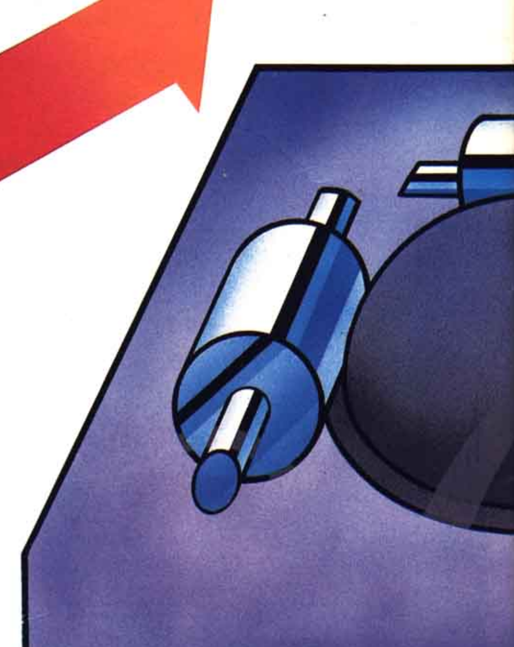
Por último, un conjunto de dos LEDs (Diodos emisores de luz) y dos fotodiodos, colocados entre las ranuras de cada disco, se encargan de traducir el movimiento de giro de éstos en una serie de impulsos eléctricos.

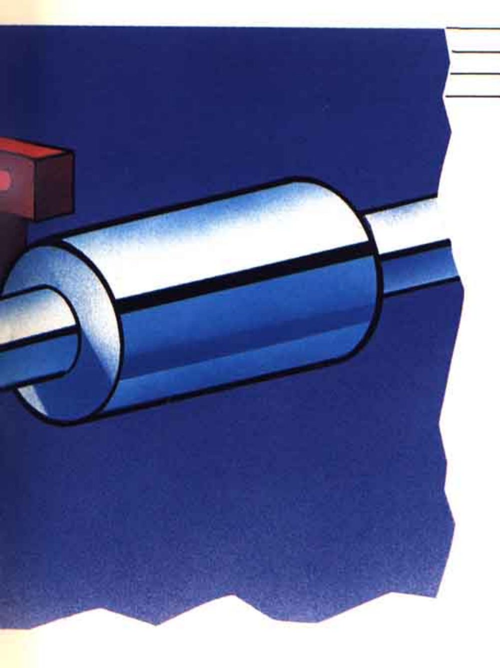
Al girar, las ranuras de los discos interrumpen o dejan el paso libre al haz luminoso que va de LED a fotodiodo. De esta forma, en el fotodiodo se genera una señal eléctrica de impulsos, en la que se encuentra codificada la información sobre el movimiento del ratón.

Un conjunto de circuitos electróni-

Al desplazar el ratón en diagonal giran ambos rodillos.







Con el giro de cada rodillo, se interrumpe periódicamente el haz luminoso.

cos, que se localizan en la misma caja del ratón, se encargan de procesar la señal eléctrica de impulsos, diciéndole al ordenador, a través del conector del *joystick*, si el ratón se ha movido hacia arriba, hacia la izquierda, en diagonal, etc.

En algunos ratones, los discos en lugar de llevar ranuras o ventanitas, llevan una serie de puntos dibujados con pintura reflectante.



Al desplazar el ratón arriba, abajo, derecha o izquierda, se produce el giro del rodillo correspondiente.

En este caso, el fotodiodo sólo recibirá luz cuando delante del LED haya un punto reflectante. Se trata de una pequeña diferencia de diseño sin mayor importancia para el usuario.

VENTAJAS DEL RATON

Este simpático accesorio se está convirtiendo en un elemento muy popular en las nuevas generaciones de ordenadores y ello por una serie de ventajas que lo situan por delante de *joysticks*, *paddles*, tabletas gráficas y otros periféricos similares.

Algunas de ellas son ventajas de tipo tecnológico. El sistema de traducir el movimiento mediante impulsos luminosos, elimina el rozamiento y el desgaste que se produce con interruptores y potenciómetros.

Ello se traduce en una mayor duración de los componentes, en menos fallos de funcionamiento y en una mayor precisión y suavidad en el movimiento del cursor por la pantalla.

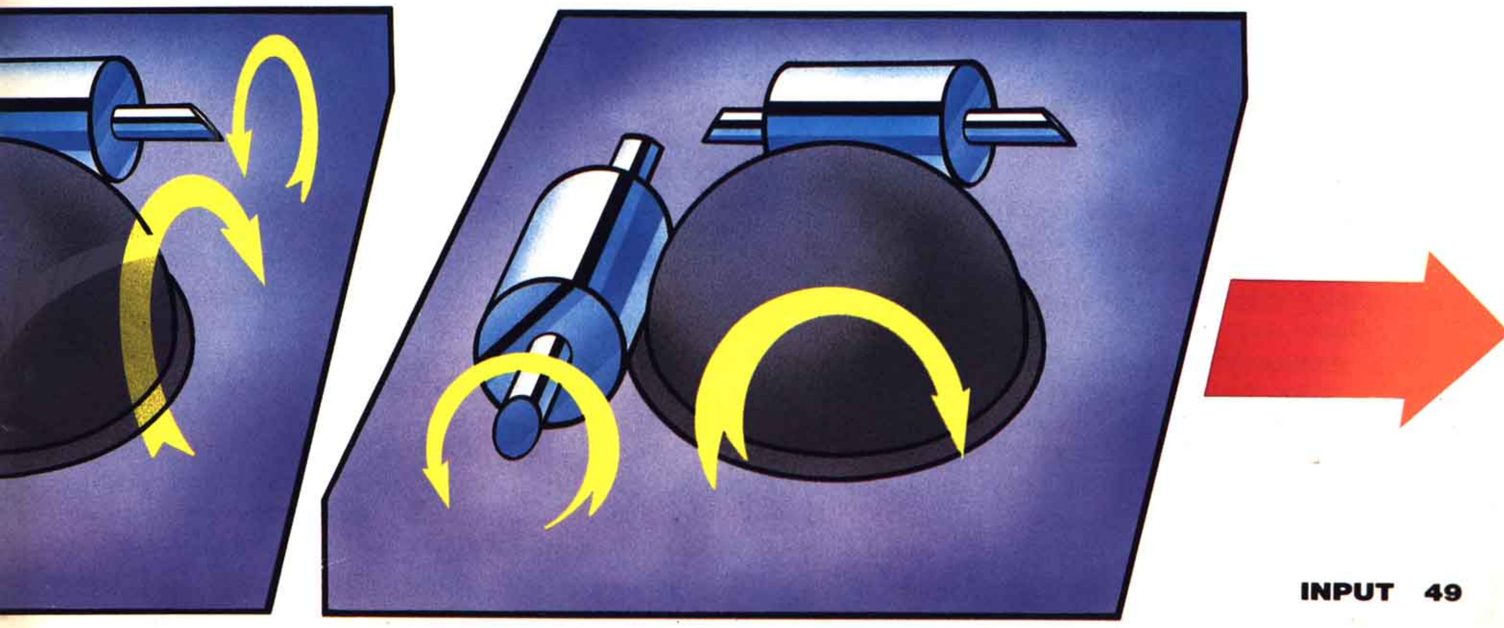
Desde el punto de vista del usuario, las ventajas son sobre todo las de comunicarse con el ordenador de una forma muy agradable, precisa y directa. Es un tanto difícil explicarlo pero el manejo del ratón es sobre todo intuitivo, no requiere apenas esfuerzo por parte del usuario, que puede concentrarse más fácilmente en la pantalla de su monitor.

Esta posibilidad se agradece sobre todo en programas como los de diseño gráfico, en los que lo esencial es la imagen que aparece sobre la pantalla. En este caso, el ratón se muestra claramente superior a los *joysticks* e incluso a las famosas tabletas gráficas, al permitir que el dibujante se concentre exclusivamente sobre el dibujo de la pantalla, y no en el teclado, en el *joystick* o en la tableta gráfica. En este caso el ratón es el elemento ideal de comunicación con el programa, pues lo que se busca es señalar puntos concretos o zonas de una imagen, y una forma de poder desplazarse rápidamente de unos puntos a otros y de unas a otras zonas. Con un ratón, el usuario sólo tendrá que hacer un ligero movimiento con la mano, sin dejar de mirar a la pantalla, para desplazar el cursor hasta la posición deseada.

En este tipo de situaciones, en las que lo más importante es desplazarse de unas zonas a otras de la pantalla, el ratón resulta muy adecuado.

Cada día son más los programas que se diseñan pensando en un ratón. En ellos el usuario se mueve a través del programa seleccionando opciones en menús gráficos o icónicos (donde cada opción posible aparece representada por un pequeño dibujo) que aparecen repartidos entre una o varias ventanas sobre la pantalla.

El futuro parece bastante claro para los ratones que van a dejar de ser únicamente unos bichos capaces de provocar el terror de muchas mujeres.



EL MAPA DE MEMORIA DE TU MSX

Un mapa de memoria no es más que una descripción de la estructura y organización de la memoria del ordenador. Conocer al detalle cómo es el mapa de memoria, es algo, además de interesante, esencial para todos los que estéis pensando en programar en código máquina. En INPUT nos hemos propuesto explicaros, a través de una serie de artículos que irán apareciendo en números sucesivos, cómo es exactamente el mapa de memoria de vuestro MSX.

En este primer artículo, vamos a contaros cual es la organización general de la memoria de un **MSX**, donde están la RAM y la ROM, y cómo se consigue mediante la división de la memoria en segmentos y páginas, una capacidad de memoria teórica de hasta ¡1 Megabyte!

Pero antes que nada vamos a recordar algunos conceptos como: memoria, direcciones de memoria, direcciones de entrada/salida, RAM y ROM.

Podemos imaginarnos la memoria como un conjunto de casilleros o cajas (las posiciones de memoria) a los que acude la CPU para dejar (escribir) o recoger (leer) todo tipo de datos.

Cada posición de memoria está formada por un conjunto de 8 bits; lo que se llama un byte. En cada byte se puede almacenar un número comprendido entre 0 y 255 (00000000 y 11111111 en binario).

Para distinguir unas posiciones de memoria de otras se numeran consecutivamente con los números 0, 1, 2, 3... etc. Estos números se llaman direcciones de memoria.

Las direcciones de entrada/salida son también un conjunto de posiciones de 8 bits, numeradas consecutivamente como 0, 1, 2, 3... etc, que normalmente se asignan a los registros de los *chips* de vídeo, *chips* de sonido y otros *chips* de entrada/salida del ordenador.

La memoria que hay en las distintas direcciones de memoria puede ser RAM o ROM. En la memoria RAM se puede leer y escribir, es decir, podemos ir a una dirección de RAM, leer el valor que contiene y, si queremos, cambiarlo por otro valor.

En cambio en la memoria ROM no podemos escribir, sólo podemos leer lo que otros escribieron en el momento de fabricación de la ROM.

El mapa de memoria nos va a decir qué direcciones corresponden a la RAM, cuáles a la ROM y cuáles a los registros de los *chips*.

MAS DE 64K

Todos los *chips* CPU tienen una serie de terminales dedicados a la tarea de direccionar la memoria. Cuantos más terminales, más direcciones de memoria podrá manejar la CPU. El número de direcciones de memoria que puede manejar una CPU con n terminales de direccionamiento es de: 2 elevado a n.

Por ejemplo, imaginemos una CPU con un sólo terminal. Como el terminal sólo puede estar a cero o a uno, sólo permite distinguir entre 2 direcciones de memoria (2 elevado 1 = 2). Con dos terminales se podrían distinguir 4 direcciones de memoria (2 elevado a 2 = 4). Uno de los *chips* de más éxito actualmente, el **Motorola 68000**, incorporado en ordenadores como el **Apple Macintosh**, el **Atari 520ST** o el **Commodore Amiga**, tiene 24 terminales, lo que le permite direccionar directamente nada menos que ¡16 Mbytes! (2 elevado a 24 = 16777216).

La CPU **Z80A**, que es la que lleva cualquier **MSX**, es más modesta.

Con sus 16 terminales de direccionamiento, no es capaz de manejar directamente más que 64K (2 elevado a 16 = 65536).

Alguno se estará preguntando: ¿Entonces, cualquier ordenador que utilice la CPU **Z80A** no puede tener más que 64K de memoria? ¿Cómo es posible entonces que existan ordenadores **MSX** con más de 64K?

Lo que hemos dicho es que la CPU **Z80A** sólo puede direccionar, directamente, 64K. Pero existen métodos indirectos para direccionar más memoria.

El estandar **MSX** utiliza uno de estos métodos para conseguir capacidades de memoria teóricas de ¡hasta 1 Megabyte!

SEGMENTOS Y PAGINAS

El estandar **MSX** define una división de la memoria en segmentos y páginas. Para aclarar conceptos conviene fijarse en la **figura 1**.

Cada segmento o *slot* puede albergar hasta 64 Kbytes de memoria, ya sea RAM o ROM, repartidos en cuatro páginas de 16K. Las páginas van numeradas de cero a tres y ocupan las siguientes direcciones dentro del *slot*: página 0 (de &H0000 a &H3FFF), página 1 (de &H4000 a &H7FFF), página 2 (de &H8000 a &HBFFF) y por último la página 3 (de &HC000 a &HFFFF).

El número mínimo de *slots*, según el estandar, es de dos; pero actualmente la mayoría de los fabricantes incluyen cuatro. Estos cuatros *slots* ofrecen capacidad para 256 Kbytes, a repartir entre RAM y ROM. La capacidad mencionada de 1 Mbyte se obtiene al ampliar cada *slot* con tres *slots* más, lo que nos da 16 *slots* de 64K cada uno, en total 1 Mbyte, pero en este tema no vamos a entrar, al menos por el momento.

Hemos dicho que la CPU **Z80A** sólo puede manejar 64K a la vez, es decir 4 páginas de 16K. Pues bien, lo más interesante del estandar **MSX** es que estas 4 páginas pueden estar en *slots* diferentes y además, en cualquier momento, es posible cambiar la asignación de cualquiera de las 4 páginas, de un *slot* a otro. Es decir, la memoria se puede configurar a voluntad cuando se quiera.

Observa la **figura 2** en la que hemos representado algunos ejemplos. Hemos partido de un **MSX** teórico con cuatro *slots*, en el que sólo se utilizan los tres primeros. El primero de todos, el *slot* 0, contiene 32K de ROM (las dos primeras páginas) con el sistema operativo y el BASIC. Los otros 32K (páginas 2 y 3) hemos supuesto que contienen RAM.

Este *slot* es el más importante en cualquier **MSX**. Se le conoce con el nombre de *slot* del sistema (*System Slot*). Por su importancia volveremos a hablar de él más adelante. El siguiente *slot* de nuestro sistema, el 1, está ocupado sólo en parte. Las páginas 0 y 3 están vacías y en las páginas 1 y 2 hemos colocado 32K de ROM (16K para el *interface* de la unidad de *diskettes* y otros 16K para un cartucho de aplicaciones).

Por último, en el *slot* 2, hemos colocado 4 páginas de RAM, lo que nos da una zona de 64K de memoria para el usuario.

Con esta configuración de memoria

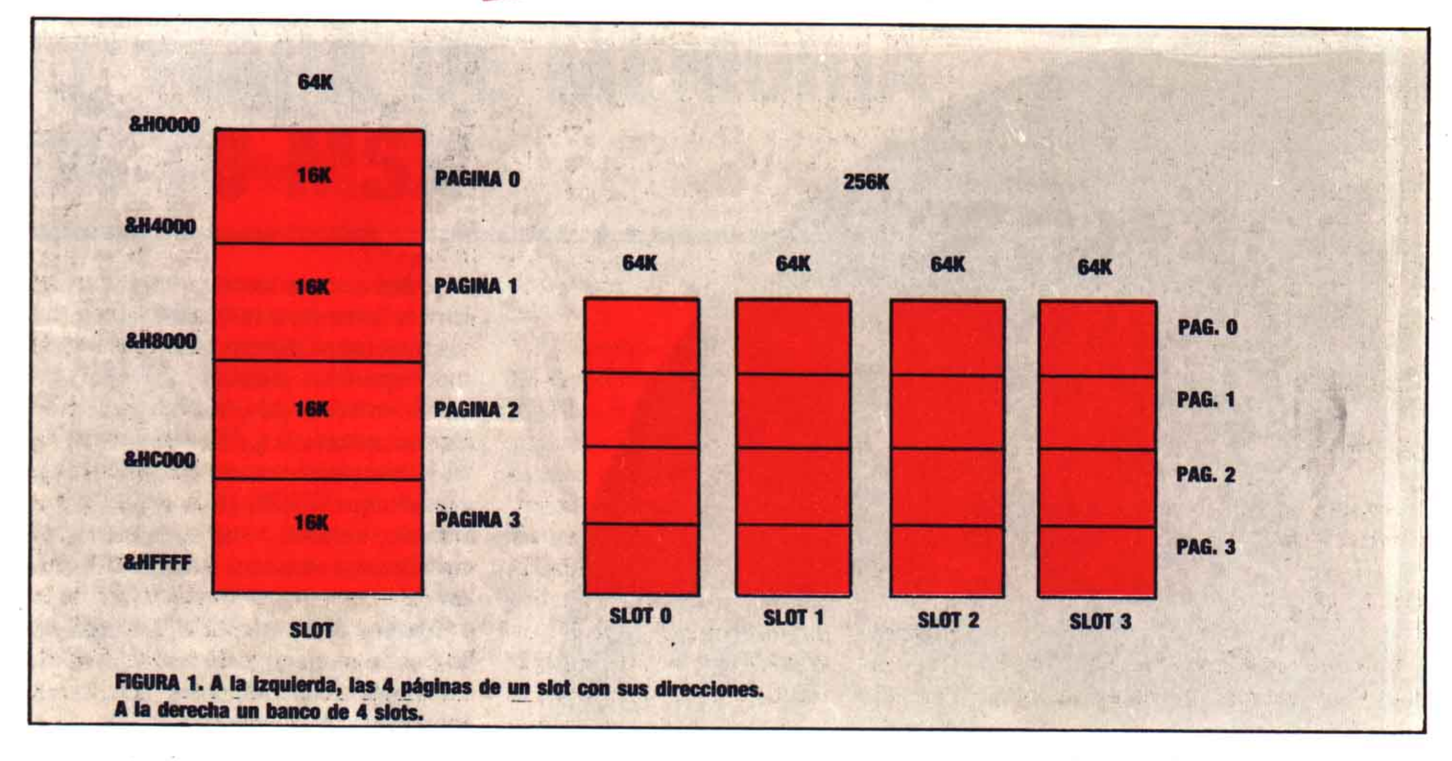


FIGURA 1. A la izquierda, las 4 páginas de un slot con sus direcciones. A la derecha un banco de 4 slots.

(se trata de un caso real, que puede darse en muchos **MSX**) y teniendo en cuenta que la CPU **Z80A** no puede manejar a la vez más que 64K, es decir 4 páginas de 16K, hemos representado tres de las muchas configuraciones que pueden obtenerse.

En el caso A, estamos trabajando con las 4 páginas del *slot* 0. Nos encontramos en el BASIC y disponemos de 32K de memoria RAM para nuestros programas (algo menos en realidad, ya que parte de esta memoria RAM la utiliza el sistema operativo, lo que nos deja únicamente con unos 28K libres).

En cualquier momento, siempre que sepamos cómo hacerlo, podremos pasar a la configuración B. En este caso también hay 32K de ROM y 32K de RAM, pero una de las páginas de ROM y las dos de RAM ya no pertenecen al *slot* 0.

Si lo deseamos, podemos cambiar a la confiuración C. En este caso tenemos a nuestra disposición 64K de RAM repartidos entre los *slots* 0 y 2. El problema es que ya no disponemos del BASIC, ni siquiera del sistema operativo.

Estos ejemplos, que muestran como se organiza la memoria de los **MSX**, merecen algunos comentarios:

Primero, trabajar en BASIC supone una importante limitación en la cantidad de memoria que podemos utilizar. Sólo podemos acceder a unos 28K.

Sin embargo, utilizando código máquina o una combinación de código máquina y BASIC, tendremos acceso a la totalidad de la memoria de que disponga nuestro **MSX**.

En segundo lugar, que la organización en segmentos y páginas, ofrece una flexibilidad muy superior a la organización convencional.

Por último hay que tener en cuenta que aunque podemos escoger páginas de distintos *slots*, siempre tendremos que escoger una página 0, una página 1, una página 2 y una página 3, es decir, lo que no podremos hacer es escoger, por ejemplo, la página 1 del *slot* 0 y la página 1 del *slot* 2, son dos páginas 1 que nunca podrán formar parte simultaneamente de los 64K a los que accede la CPU.

COMO CAMBIAR DE CONFIGURACION

La configuración de la memoria, es decir, la asignación de las distintas páginas de los distintos *slots*, la controla el *port* A del *chip* **PPI 8255**. Pronto dedicaremos un artículo a comentar el funcionamiento y las aplicaciones de este *chip*. Por el momento nos basta con saber que el *port* A del *chip* es una especie de registro de 8 bits que responde a la dirección &HA8 de la zona de entrada/salida de la CPU. Cada pareja de bits de este *port* (el 0 y el 1, el 2 y el 3, etc), define a cual de los cuatro *slots* pertenece cada una de las páginas. En la **figura 3** hemos representado los ocho bits del *port*, una tabla con la correspondenica entre el valor de los bits y la asignación de páginas y un ejemplo concreto de asignación.

Podemos leer el valor de este *port* A desde código máquina o desde BASIC. En el primer caso utilizaremos la instrucción IN, en cualquiera de sus modalidades de direccionamiento, mientras que desde BASIC haremos uso de la instrucción equivalente INP. Por ejemplo, prueba a teclear:

```
PRINT BIN$(INP(&HA8))
```

Con ello obtendrás el valor del *port* A, o lo que es lo mismo, la asignación de páginas que en ese momento hace tu ordenador. Supongamos que obtienes lo siguiente:

```
10 10 00 00
```

Esto quiere decir que las páginas 0 y 1 pertenecen al *slot* 0 (00 en binario) mientras que las páginas 2 y 3 son del *slot* 2 (10 en binario).

Para modificar el valor del *port* A y reconfigurar la memoria, podemos de nuevo elegir entre código máquina, con la instrucción OUT, o BASIC, también con la instrucción OUT. Por ejemplo:

```
OUT &HA8,00
```

asignará las cuatro páginas al *slot* 0, mientras que:

```
OUT &HA8,A0
```

hará que la CPU trabaje con las páginas 0 y 1 del *slot* 0 y las páginas 2 y 3 del *slot* 2.

Ten en cuenta que al utilizar la instrucción OUT desde BASIC para modificar el *port* A, puedes quedarte colgado. Esto ocurrirá siempre que tomes las páginas 0 y 1 de otro *slot* que no sea el 0 (lo que habrás hecho es desconectar la ROM).

Lo que si puedes hacer es llamar, desde un programa BASIC, a una rutina en lenguaje máquina que se encargue de cambiar de configuración, por ejemplo sustituyendo las páginas de ROM por páginas de RAM.

A partir de ese momento, la rutina en lenguaje máquina podrá trabajar en la nueva zona de RAM, haciendo uso de todas las posiciones de memoria que ahora tiene a su disposición. Al terminar, la rutina tendrá que volver a la configuración inicial, conectando de nuevo la ROM y el BASIC, antes de devolver el control al programa BASIC original. Siempre que al pasar a una nueva configuración te ocupes de que no se modifique la zona de memoria en la que se almacena el programa BASIC, podrás volver a él y te lo encontrarás intacto y funcionando perfectamente.

Un detalle curioso: la rutina en lenguaje máquina que cambia la configuración de la memoria, podrá cambiar todas las páginas menos la página en la que ella misma se encuentra. Por ejemplo, supón que la rutina en lenguaje máquina se encuentra colocada en la página 3 del *slot* 0. Si al cambiar de configuración se asigna la página 3 al *slot* 2, nos encontramos con que la propia rutina en lenguaje máquina ha desaparecido ya que, aunque la rutina sigue estando en la página 3, está en la página 3 de un *slot* que no está conectado.

También se puede hacer algo curioso para no perder el BASIC al cambiar de configuración. Se trata de copiar toda la ROM, de las páginas 0 y 1 del *slot* 0, y transferirla a una zona de RAM que ocupe también las páginas 0 y 1, pero, por ejemplo, en el *slot* 2. De esta forma podemos cambiar la configuración sin perder el BASIC. Es más, como hemos copiado la ROM en una zona de RAM, podremos modificar todas las rutinas que constituyen el BASIC y el sistema operativo y que antes, al estar en ROM, no podíamos modificar.

Todas estas posibilidades que comentamos y muchas más, dan una idea de la enorme flexibilidad que proporciona la organización de la memoria en segmentos y páginas. En próximos capítulos os daremos los listados para que podáis experimentar con los cambios de configuración y con las modificaciones de las rutinas de la ROM.

Hay que tener en cuenta que todas estas posibilidades van a depender de la cantidad de memoria que tenga vuestro modelo de **MSX** y que lo que hemos mencionado es de aplicación inmediata sólo en aquellos **MSX** que tengan RAM en algún *slot* distinto del 0. Esto ocurre en los **MSX** de 64K de RAM, o en los que teniendo 16K de RAM, llevan acoplados cartuchos de expansión de memoria.

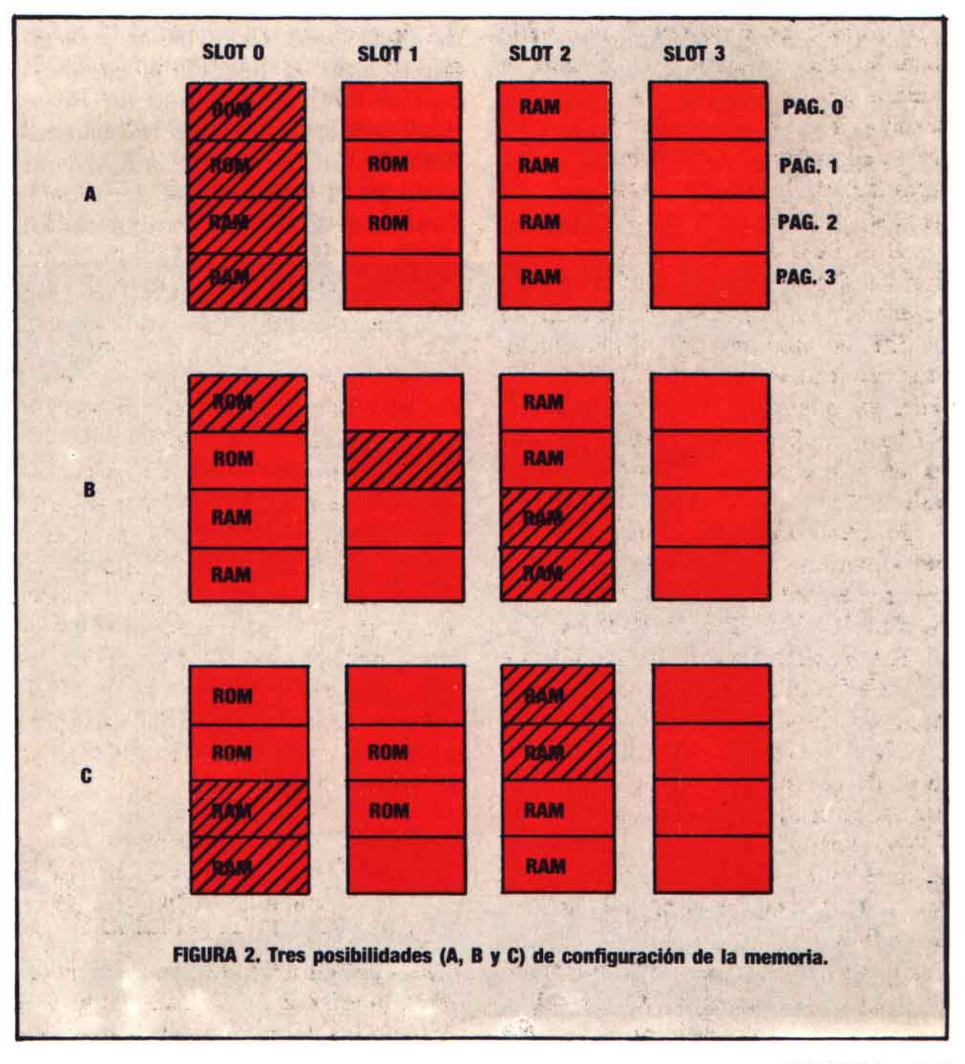


FIGURA 2. Tres posibilidades (A, B y C) de configuración de la memoria.

EL SLOT 0

Cuando encendemos un ordenador **MSX**, las páginas 0 y 1 quedan automáticamente asignadas al *slot* 0, o *slot* del sistema. En este *slot* y en las citadas páginas hay 32K de memoria ROM con el sistema operativo y el intérprete de lenguaje BASIC.

De esta forma, al encender el ordenador las rutinas del sistema operativo entran en funcionamiento, se hacen con el control y llevan a cabo las acciones necesarias para que podamos empezar a trabajar.

La situación y el contenido de estas dos páginas del *slot* 0 es idéntica para todos los ordenadores **MSX**, y es una de las claves de la compatibilidad entre distintos modelos.

El resto de las páginas de los distintos *slots* puede ya variar de unos modelos a otros.

Por ejemplo, si consideramos el *slot* 0 del modelo **HB-55P** de **Sony**, nos encontramos un programa de utilidades en ROM: el ***Personal Data Bank***.

La página 3 es la única que contiene RAM, por lo que este sistema, a menos que se añada un cartucho de expansión de memoria RAM, sólo ofrece 16K de memoria para el usuario.

Otro modelo de **Sony**, el **HB-75P**, tiene la misma configuración de memoria en el *slot* 0, pero en cambio ofrece 4 páginas de RAM (64K) situadas en el *slot* 2. Cuando se accede al BASIC en este modelo, las páginas 2 y 3 quedan asignadas al *slot* 2, con lo que el usuario dispone de 32K para sus programas BASIC.

EL MAPA DE LA RAM

Aunque en próximos números comentaremos en detalle el mapa de memoria RAM y las direcciones de interés para el programador, vamos a dar aquí un mapa a grandes rasgos de la organización de esta memoria.

Como puede verse en la **figura 4**, la RAM comienza en la página 2, es decir en la dirección &H8000, y se extiende hasta el final de la página 3, esto es hasta la dirección &HFFFF. Esto es cierto salvo en los sistemas **MSX** de 16K, en los que la RAM empieza en la página 3, es decir en la dirección &HC000, extendiéndose también hasta la dirección &HFFFF.

Hay que tener en cuenta que, según el modelo de **MSX** que manejemos, estas dos páginas de RAM pertenecerán a uno u otro *slot*.

Pero al trabajar en BASIC este detalle no tiene ninguna importancia, lo único que nos importa es saber que la RAM ocupa las páginas 2 y 3 (sólo la 3 en los **MSX** de 16K) y que responde a las direcciones citadas.

Esta RAM se organiza en dos zonas bien diferenciadas. Una de ellas es la zona de trabajo del sistema. Se encuentra a partir de la dirección &HF380 y ocupa hasta el final de la RAM, hasta la dirección &HFFFF.

Esta zona la utiliza el sistema operativo para muy distintas funciones.

Por ejemplo, es evidente que al encender el ordenador, nos encontramos en el modo de pantalla 0 y las teclas de función llevan asignados los comandos que aparecen en la línea inferior de la pantalla. Estas son informaciones que mantiene el sistema operativo en la zona de RAM del sistema. Así, cuando cambiamos el contenido de las teclas de función, se altera parte de la RAM del sistema para reflejar el cambio.

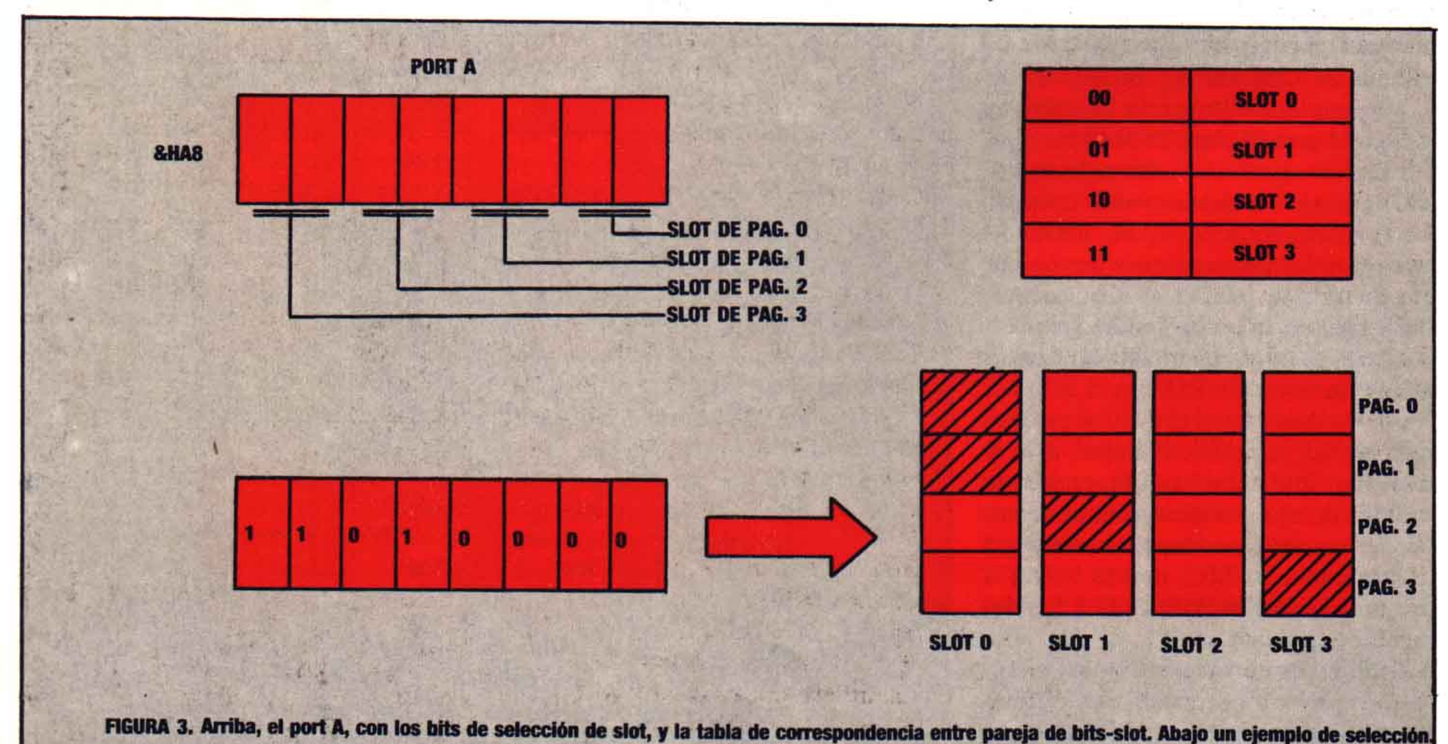


FIGURA 3. Arriba, el port A, con los bits de selección de slot, y la tabla de correspondencia entre pareja de bits-slot. Abajo un ejemplo de selección.

Estos y otros muchos valores que definen el modo de funcionamiento de la máquina, se almacenan en la memoria RAM del sistema, entre las direcciones &HF380 y &HFD99.

Actuando con PEEK y POKE o desde rutinas en lenguaje máquina sobre estas direcciones, se pueden conseguir efectos muy interesantes.

En un próximo capítulo comentaremos este aspecto y os diremos cuales son las direcciones más interesantes de esta zona de RAM.

Por otro lado, entre las direcciones &HFD9A y &HFFC9, nos encontramos con lo que se denominan vectores del sistema. Son direcciones de RAM a las que acceden deteminadas rutinas de la ROM. Normalmente, estos vectores contienen, todos ellos, el valor &HC9, que corresponde a la instrucción RET, algo así como RETURN, pero en código máquina. De esta forma, cuando desde la ROM se salta a uno de estos vectores se produce inmediatamente un retorno a la ROM, y no pasa nada. Sin embargo, si cambiamos el valor de estos vectores, podemos desviar el flujo normal de ejecucción hacia rutinas creadas por nosotros. Esto nos va a permitir algo así como interceptar al BASIC y modificar el funcionamiento del mismo.

Por ejemplo podemos interceptar la rutina de la instrucción INPUT (a través del vector que comienza en &HFDEO) y eliminar el símbolo de interrogación de dicha instrucción, o sustituirlo por cualquier otro símbolo que se nos ocurra.

También en un próximo capítulo daremos una lista de los vectores más interesantes y de como manejarlos.

La otra zona de memoria RAM, tan importante o más que la RAM del sistema, es la RAM de usuario, en la que se almacenan los programas BASIC, las rutinas en código máquina, y en general, todos los datos que manejan los programas.

Esta zona empieza en la dirección &H8000, al principio de la memoria RAM, y se extiende hasta &HF380, donde empieza la RAM del sistema.

Los programas y las variables BASIC se almacenan en esta zona de la RAM con la siguiente estructura. A partir de la dirección más baja, se almacena el programa BASIC. El mes pasado publicamos un artículo sobre el formato de almacenamiento de los programas. Después del programa, nos encontramos con la zona de almacenameinto de las variables BASIC. A continuación se almacenan las matrices que incluye el programa. A medida que crece un programa, o si incluye más variables o más matrices, estas tres zonas de almacenamiento ocuparán más memoria y crecerán hacia el final de la RAM.

Por otro lado, desde la dirección &HF380 y hacia el comienzo de la RAM, nos encontramos con el bloque de control de ficheros, que se utiliza en las operaciones de entrada y salida de datos hacia los ficheros en *cassette* o *diskette*. A continuación viene una zona en la que se almacenan las cadenas de caracteres.

Después nos encontramos con la pila del sistema, en donde se almacenan las direcciones de retorno de las instrucciones NEXT y RETURN.

Entre el final de esta zona y el final de la zona de almacenamiento de matrices, queda una zona de RAM libre, que, eso sí, se irá haciendo más pequeña a medida que crezcan los programas BASIC.

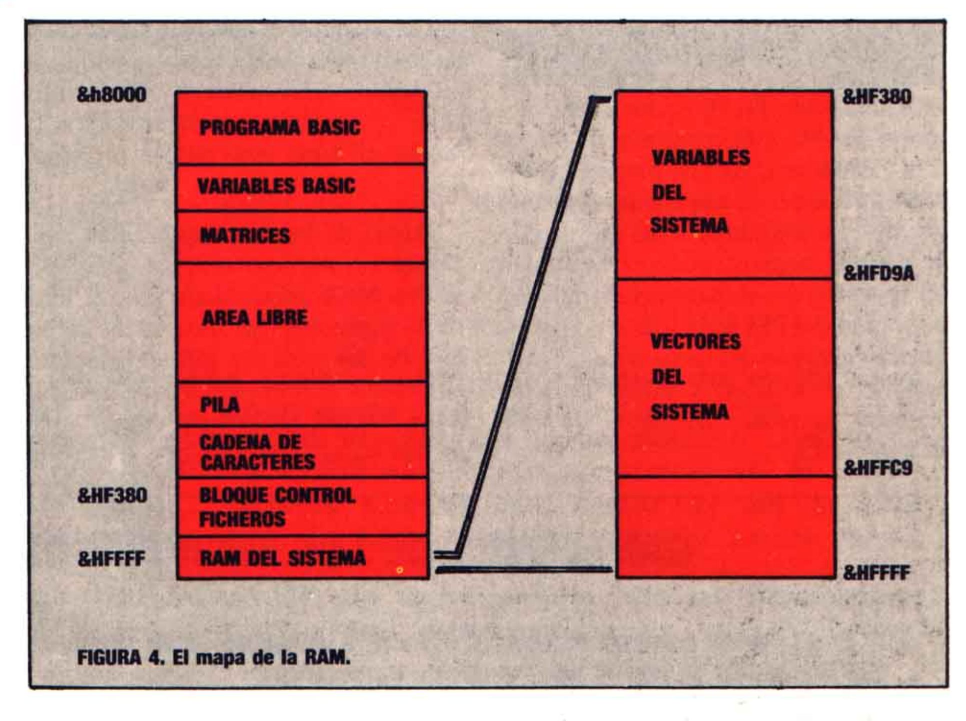


FIGURA 4. El mapa de la RAM.

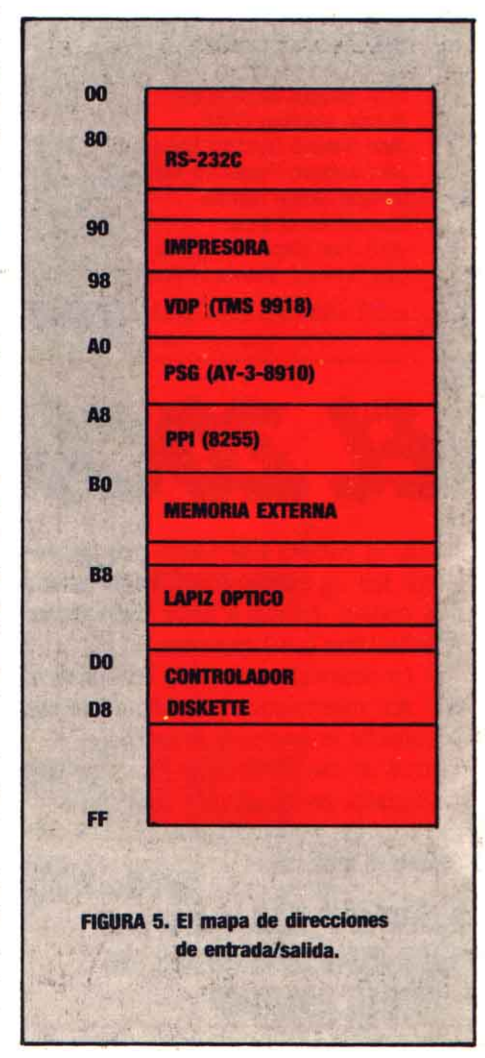


FIGURA 5. El mapa de direcciones de entrada/salida.

EL MAPA DE ENTRADA/SALIDA

Además de las direcciones de memoria RAM o de memoria ROM, la CPU **Z80A** es capaz de manejar otras 256 direcciones. Estas son las direcciones de entrada/salida a las que, como ya hemos comentado, corresponden los registros del *chip* de vídeo, del *chip* de sonido del **PPI 8255** y de otros *chips* que se encargan del control de las operaciones de entrada y salida de datos (teclado, monitor, impresora, *cassette*, *diskette*, etc).

En la **figura 5** hemos representado un mapa de cómo están asignadas las diferentes direcciones a los distintos *chips*. En próximos artículos estudiaremos el tema con mayor profundidad.

Antes de terminar queremos mencionar un aspecto muy característico de los **MSX** relacionado con el tema de la memoria. Se trata nada menos que de un total de 16K de memoria RAM que constituyen lo que se denomina **VRAM** (RAM de vídeo). Esta memoria, al contrario que en la mayoría de los ordenadores domésticos, no pertenece a la memoria RAM de usuario, sino que se sitúa al margen de la zona normal de direcciones, controlada de forma independiente por el *chip* de vídeo. Es decir, no pertenece al espacio de direcciones de la RAM, o lo que es lo mismo, no le «roba» RAM al usuario. Por esta razón no hemos hablado de ella en este artículo. Nos ha parecido más interesante posponer su estudio para más adelante, y dedicarle unas cuantas páginas en exclusiva.

GANADORES DE LOS MEJORES DE INPUT MSX

En el sorteo correspondiente al número 1 realizado entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Gabriel Martos Expósito	Ubeda (Jaén)	Yie ar Kung Fu
Javier Cabrera López	Estepona (Málaga)	Profanation
Enric Nogues Montuenga	Reus (Tarragona)	Turmoil
Vicente Martínez Gosp	Pego (Alicante)	Turmoil
José Antonio Sánchez Casales	Elche (Alicante)	Alien 8
José Antonio Fernández Márquez	Badalona (Barcelona)	Soccer
Enrique Bretón Montiel	Logroño	Ghostbusters
Germán Verea Jerez	La Herradura (Granada)	Profanation
José Juan Requena Aguado	Elda (Alicante)	Knigth Lore
Luis Santiago Vilanova Piñeiro	Lugo	Hyper Sports III

EL ZOCO DE INPUT



Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:

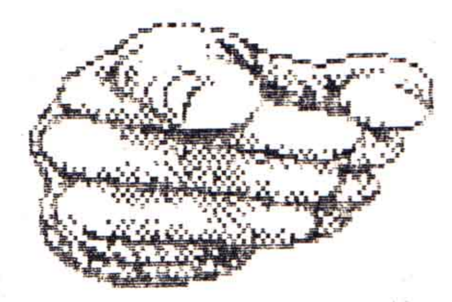
a) La propuesta tendrá que ver con la microinformática.

b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.

Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

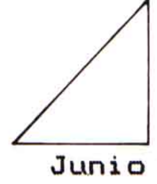
INPUT MSX ZOCO
c/. Alberto Alcocer, 46
28016 MADRID

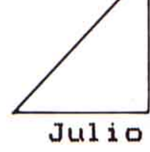
TU  PUEDES SER EL AFORTUNADO

que reciba gratuitamente en su casa durante un año todos los programas que sean lanzados por Erbe o U.S. Gold para los ordenadores MSX.
Podrás participar en el sorteo con solo enviarnos esta página una vez que hayas pegado las cuatro esquinas coloreadas que aparecerán en los tres números siguientes de INPUT. Además tienes que recortar de alguna carátula de un *cassette* los anagramas de Erbe y U.S. Gold y pegar en los lugares indicados.
La fecha tope de participación es el 30 de Septiembre. En el número de INPUT de octubre se publicará el nombre del ganador.

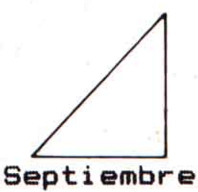
Datos del concursante:
Nombre:...
1er. Apellido:..................................
2o. Apellido:...................................
Año de nacimiento:...........
Dirección:.......................... No.:......
Ciudad:................. Provincia:............
Dto.Postal:............. Teléfono:............

Nota: Si bien esta página puede ser fotocopiada o copiada a máquina, no entrarán en el concurso aquellas a las que falte alguno de los seis recortes exigidos o que estos sean copia del original.


Junio


Julio


Extra de verano


Septiembre





Junio



PUESTO	TITULO	PORCENTAJE
1.º	H.E.R.O	17,8 %
2.º	Soccer	14,7 %
3.º	Knight Lore	14,2 %
4.º	Alien 8	10,7 %
5.º	Hyper rally	9,5 %
6.º	Hyper sport	8,3 %
7.º	Profanation	8,3 %
8.º	Road fighter	5,9 %
9.º	Yie ar kung-fu	5,9 %
10.º	Zakil Wood	4,7 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Junio de 1986



LA LAMPARA MAGICA

DATOS GENERALES

TITULO Master of the Lamps

FABRICANTE Activisión

CLASE DE PROGRAMA Juego

FORMATO Cassette

CALIFICACION (Sobre 10 ptos.)

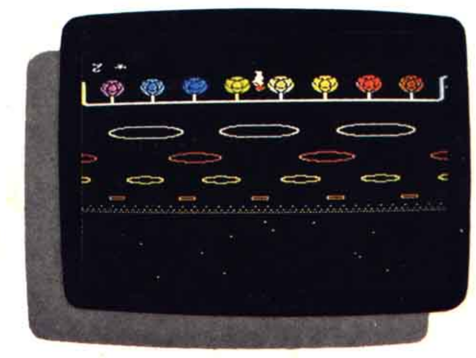
ORIGINALIDAD	**9**
INTERES	**8**
GRAFICOS	**8**
COLOR	**9**
SONIDO	**8**
TOTAL	**42**

Una serie de espíritus malignos se han apoderado del reino y han sembrado el terror. Nuestro objetivo será ayudar al joven principe a recuperarlo llegando hasta los espíritus y superando las pruebas para ir completando las lámparas de siete piezas, hasta llegar al grado de Maestro de Lámparas y recuperar el trono del reino.

Para llegar al escondite de los espíritus malignos debemos montar en nuestra alfombra mágica y conducirla a través del tunel de entrada, con cuidado de no salirnos de los límites.

Cuando lleguemos al escondite de los espíritus les avisaremos golpeando tres veces el gong. Superaremos las pruebas golpeando los gongs de colores en la secuencia pedida por el espíritu, lo más rápidamente posible, ya que si no, seremos transportados otra vez al comienzo del tunel de entrada, debiendo empezar de nuevo el recorrido.





Cada gong se corresponde con un tono y un color, y debemos tener en cuenta que el espíritu no siempre nos dará las dos pistas.

El juego nos ofrece tres niveles:

MAGIC CARPET: Para practicar vuelos de prueba por los túneles.

SEVEN TRIALS: Más sencillo. Al completar una de las lámparas de siete piezas adquirimos el grado maestro.

THRONE QUEST: Para llegar al grado de maestro deberemos completar tres lámparas de siete piezas.

Master of the lamps es un juego muy entretenido, que aprovecha muy bien las posibilidades gráficas, de color y de sonido de nuestro ordenador.

Hay que destacar el efecto conseguido por el vuelo con la alfombra a través de los túneles, que nos sorprenderá por su realismo.

El mapa de KNIGHT LORE

El **Knight Lore** es un viejo conocido de todos los aficionados a la microinformática. Su primera versión, realizada en 1984 para **Spectrum**, cayó como una bomba entre los caducos esquemas entonces imperantes, y sentó las bases de un nuevo estilo. Por todo ello, **Knight Lore** está unánimemente considerado como uno de los «clásicos» de la programación.

TE LO CONTAMOS

Se trata de una aventura con un planteamiento bastante sencillo. Consiste en encontrar los catorce ingredientes de una pócima que permitirá al atormentado protagonista liberarse del hechizo que le convirtió en hombre-lobo por el resto de sus días. El problema es que hay que buscarlos por un laberinto de más de cien habitaciones, con innumerables peligros acechando, y sufriendo espectaculares transformaciones de hombre a lobo y de lobo a hombre, cada noche y cada amanecer. Además, dichos ingredientes deben ser introducidos en un caldero mágico, custodiado por un mago y uno de sus conjuros, **en el orden correcto**, pues de lo contrario la pócima no surte el menor efecto. Tal «orden correcto» no es siempre el mismo, como tampoco se encuentran siempre los mismos objetos en los mismos lugares, pero existen indicios, de los que hablaremos más adelante, que permiten neutralizar estos inconvenientes.

Se inicia el juego con cinco preciosas vidas, y a lo largo del mismo podemos encontrar en algunas habitaciones pequeñas estatuillas que valen por una «vida extra», hasta un total de cuatro.

La aventura comienza en una de las cuatro salidas posibles que existen, elegida de forma aleatoria. Si tomáis el mapa que acompaña a este comentario, veréis que las hemos marcado con una «S». También hemos señalado los lugares en los que podréis encontrar un objeto, con un pequeña flecha en forma de triángulo.

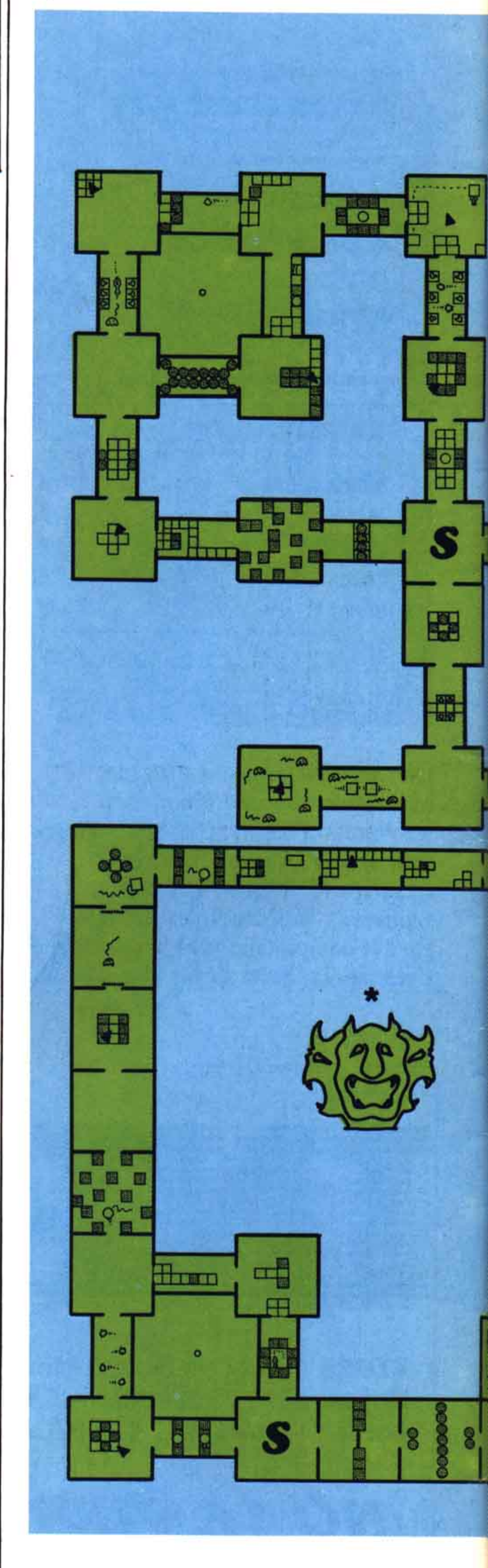
Se pueden transportar en todo momento hasta un máximo de tres objetos, que es posible recoger o dejar en cualquier parte del laberinto. Si se deja uno de ellos en una habitación, siempre se encontrará allí hasta que sea recogido de nuevo.

El tiempo tiene una importancia capital en la aventura, por dos razones: una, que sólo se dispone de cuarenta días con sus correspondientes noches para cumplir la misión, y otra, que al anochecer y al amanecer el hombre lobo interrumpirá sin previo aviso lo que esté haciendo, por delicado que sea, para sufrir su metamorfosis.

POR MAS SEÑAS

Como ya hemos dicho, la primera versión que se realizó del **Knight Lore** tiene ya una historia de casi dos años. Su éxito ha sido sabiamente aprovechado por **Ultimate** para hacer una adaptación al sistema **MSX**.

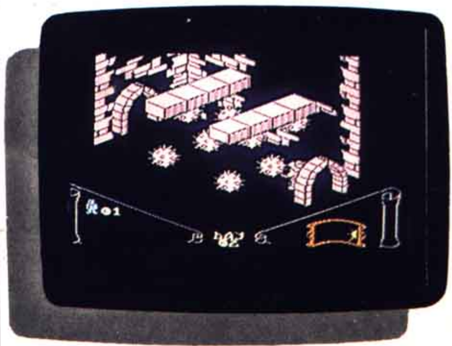
El programa utiliza un nuevo modelo de programación que sus creadores bautizaron con el nombre de **Filmation**, y que consiste en un efecto de falsa perspectiva con proyección isométrica, tan logrado que tenemos que



KNIGHT LORE

CALDERO MAGICO

MSX



palpar el frío vidrio de la pantalla para cerciorarnos de que lo que vemos tiene dos dimensiones y no tres. Técnicamente, y teniendo en cuenta las limitaciones de todo microordenador doméstico, es casi perfecto.

LO BUENO Y LO MALO

Comenzando por lo malo, diremos que el único aspecto negativo que presenta es su monocromía gráfica. Pero también debemos decir en su descargo que la razón de dicho defecto se encuentra más en las limitaciones técnicas inherentes a la microinformática que en los fallos achacables al propio programa.

En cuanto a lo bueno, destacaremos el sorprendente efecto de tridimensionalidad conseguido, la calidad de los gráficos y el movimiento del protagonista (sobre todo la metamorfosis), y los efectos sonoros (especialmente el tema musical del comienzo). Por otro lado, el interés del programa es altísimo, y seguro que os hará pasar muy buenos y largos ratos.

LA VOZ DE SU AMO

¿Cómo conocer el orden correcto en que debemos introducir los objetos



en el caldero? Muy sencillo: si prestamos atención, será el propio caldero quien nos lo diga. Cuando entremos a la habitación en que se encuentra, durante breves segundos aparecerá entre la bruma la figura del ingrediente que se necesita. Después, debemos actuar con rapidez, pues la bruma se convertirá en un chisporreteo letal que se abalanzará sobre nosotros.

Si bien los objetos no aparecen siempre distribuidos de idéntica forma, sí tenemos la seguridad de que en los lugares que hemos marcado en el mapa siempre aparecerá un objeto o una vida extra.

La táctica más adecuada es recoger todos los que encontremos y acumularlos junto a la habitación del caldero. Existen treinta y dos objetos en total, pero no son más que réplica de ocho tipos distintos, que son los siguientes: copa, frasco de veneno, bola mágica, diamante, bota, taza, botella y vida extra. Existen, por tanto, cuatro objetos de cada uno de estos tipos. Cuando necesitemos, por ejemplo, una botella, cualquiera de las cuatro que habrá repartidas por el laberinto nos servirá.

En ocasiones, los objetos también nos serán útiles para subir sobre ellos y saltar hasta lugares que, de otra forma, nos serían inaccesibles.

Algunos objetos los encontraremos «escondidos» bajo cubos que se desintegrarán al tocarlos. En estos casos, sin embargo, debemos actuar con cuidado porque también podemos encontrarnos con una mina o una plataforma de agujas.

Siempre que en una habitación nos encontremos con un peligro animado (bolas asesinas, fantasmitas, robots, etc.) lo más adecuado es movernos con la mayor rapidez.

Además, antes de emprender una acción delicada, es imprescindible cerciorarnos de que no está a punto de producirse una transformación del hombre-lobo.

VALORACION GENERAL

Después de los comentarios que hemos hecho anteriormente, el lector intuirá qué es lo que aquí vamos a decir. En efecto, **Knight Lore** es un programa como hay pocos, que merece una calificación muy alta, con una felicitación efusiva por nuestra parte a quienes, siguiendo nuestro consejo, lo compren y disfruten.

★★★★★★★★★★★★★★★★★★★★★★★★★

CINTURON NEGRO

DATOS GENERALES

TITULO Yie Ar KUNG FU 2

FABRICANTE Konami

CLASE DE PROGRAMA Juego

FORMATO Cartucho

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	9
GRAFICOS	9
COLOR	8
SONIDO	8
TOTAL	42



Con este juego vamos a vivir la aventura de **Lee Young**, el hijo de **Lee**, el gran maestro de **Kung-Fu**, que derrotó a la banda de criminales de **Chop Suey**.

Sin embargo **Yen Pei** consiguió escapar y formar, con sus secuaces y jefes militares un imperio de terror por toda China. A este malvado imperio deberá enfrentarse **Lee Young**, sin más ayuda que sus conocimientos de **Kung-Fu**... y nuestra habilidad, por supuesto.

Para lograr su objetivo, **Lee Young** deberá derrotar sucesivamente a cuatro jefes militares, cada uno de ellos con sus técnicas especiales de lucha, a cual más mortífera.

Además, para llegar a entablar combate con cada uno de estos jefes, nuestro héroe tendrá que dar cuenta de las escuadras de luchadores enanos que les dan escolta.

Hay que destacar la impresión de realismo en el movimiento de **Lee**, que puede ejecutar una gran variedad de golpes y técnicas (golpes de puño, golpes de pierna, patadas altas y bajas, patadas en salto, esquivas, etc.) mediante diversas combinaciones en las teclas del cursor o el *joystick*.

En la lucha tendremos que vigilar constantemente nuestro nivel de energía así como el de nuestro oponente. Cada golpe válido dado a nuestro oponente disminuirá su nivel de energía, y cada golpe que recibamos disminuirá el nuestro propio. El primero que agote su reserva de energía o **KI** quedará fuera de combate.

Pero **Lee Young** posee recursos secretos que desconocen sus oponentes. Cada vez que derrotemos

a una escuadra de atacantes enanos obtendremos una hoja de TE OO-LONG. Con cinco de esas hojas podremos reforzar nuestro nivel de energía.
También podremos tomar los tazones de **Chow-mein**, que nos harán invisibles durante algunos segundos, que pueden ser muy valiosos.
Yie-ar Kung-Fu 2 es un estupendo juego de artes marciales y no cabe duda de que nos hará sentirnos verdaderos maestros en el arte del **Kung-Fu**. El juego tiene además la opción de dos jugadores, en cuyo caso uno de ellos puede elegir ser uno de los malvados jefes militares, y entablar un combate personal con **Lee Young**, gobernado por el segundo jugador.
En definitiva, un juego que con su realismo y originalidad puede hacernos pasar ratos verdaderamente agradables y entretenidos.

★★★★★★★★★★★★★★★★★★★★★★★★★★

EL IMPERIO DEL DRAGON

¡Cuidado con los malvados habitantes del templo maldito! Intentarán por todos los medios deshacerse de nuestro héroe, que sólo puede defenderse con su maestría en el complicado arte oriental del **Taekwon-do**.
Sus piernas son mortíferas, pero ellos son muchos, y muy peligrosos. Nuestro héroe podrá dar cuenta con facilidad de los esbirros inferiores,



DATOS GENERALES	
TITULO	Taekwon-do
FABRICANTE	Philips
CLASE DE PROGRAMA	Juego
FORMATO	Cartucho

CALIFICACION (Sobre 10 ptos.)	
ORIGINALIDAD	9
INTERES	9
GRAFICOS	9
COLOR	8
SONIDO	8
TOTAL	43





que no saben luchar. Pero deberá tener mucho cuidado con los maestros lanzadores de puñales, a los que no es tan fácil derrotar de un sólo golpe.
De nuevo nos encontramos con otro estupendo juego basado en un arte marcial oriental: el **Taekwon-do**. Es esta una modalidad de lucha en la que las piernas juegan un papel fundamental. Esto lo han tenido muy en cuenta los creadores de este juego, en el cual nuestro protagonista puede ejecutar una gran variedad de golpes, gobernado por las teclas del cursor o por un *joystick*.
Podemos de esta forma propinar contundentes patadas agachándonos, de pie o en salto, e incluso lograr ese golpe tan espectacular que es la patada en giro, para deshacernos de varios contrarios a la vez. También podrmos agacharnos y saltar para esquivar los mortíferos puñales que lanzan los maestros.
Pero ¡cuidado!, si nos agarran los esbirros y no podemos soltarnos o nos alcanzan dos puñales seremos derrotados y caeremos sin vida al vacío.
La sensación de realismo en el movimiento del protagonista y en las técnicas que ejecuta es perfecta.
Con este estupendo juego podremos vivir interesantes aventuras, a la vez que nos haremos verdaderos expertos en el contundente arte marcial del **Taekwon-do**.

UN DIA EN EL ESTADIO







DATOS GENERALES

TITULO Hyper-Sports III

FABRICANTE Konami

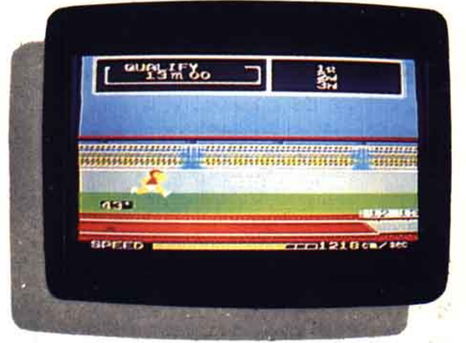
CLASE DE PROGRAMA Juego

FORMATO Cartucho

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	7
GRAFICOS	7
COLOR	8
SONIDO	8
TOTAL	37





En esta ocasión comentamos otro juego de competición deportiva. **Konami** nos ofrece una miniolimpiada en la que debemos demostrar nuestra habilidad y destreza en cuatro pruebas.
La primera de ellas es una carrera ciclista, en la que podremos medirnos con nuestro rival (que puede ser otra persona o bien el propio ordenador), tanto en velocidad como en destreza para conducirnos por el circuito, evitando los estrechamientos y adelantando a los otros corredores. No podemos dormirnos en los laureles, puesto que necesitamos hacer una marca mínima si queremos seguir jugando.
Una vez superada esta prueba, pasaremos al triple salto, en el que también tendremos que realizar una marca mínima para continuar.
Después practicaremos un deporte muy divertido. Tendremos que lanzar un disco que se desliza por el hielo y dejarlo lo más cerca posible del centro de un blanco. Para ello contaremos con dos personajes que nos ayudarán alisando y barriendo el hielo en la trayectoria del disco, para orientar su movimiento hasta el centro del blanco. Estos personajes, por supuesto, los manejaremos nosotros. Disponemos de la posibilidad de regular la temperatura del hielo. Cuanto más frío esté, más rápido se deslizará el disco.
Y por último realizaremos la prueba de salto con pértiga. Tendremos que esmerarnos, porque otra vez hay marca mínima.
Dependiendo del resulado que obtengamos en cada una de las cuatro pruebas anteriores, iremos obteniendo una serie de puntos, que iremos acumulando, para ir a la caza del *record* mundial.
El movimiento de nuestro deportista, así como su velocidad, potencia, saltos, etc., se controlan con el *joystick*, o bien con las teclas del cursor y la barra espaciadora.
Cuanto más rápido los manejemos, mas potencia y velocidad obtendremos.
Hiper-Sports se enmarca dentro de los tradicionales juegos de competición deportiva y, aunque no ofrece ninguna novedad especialmente interesante, puede hacernos pasar ratos entretenidos.
Hay que decir además, que en nuestra opinión, se podría haber mejorado un poco más la calidad de los gráficos, para dar una sensación de movimiento más real de los personajes.



EL ZOCO

Vendo HIT BIT SONY 75P MSX, 80K RAM, poco uso, con garantía por 3 meses, cables de conexión, manuales en español, 45 programas comerciales (juegos y gestión), por sólo 45.000 ptas.
Francisco Angel Mólina
Avda. de Barcelona, 326, 4°B
Tel. (958) 11 96 86 de 21 a 23 horas
18006 Granada

Cambio juegos de cinta (Les Flics, Misión de Combate, Mini-Golf...) fabricatest (Tutor) o cartucho (Code Runner) por otros juegos a convenir. Escribir a:
Federico Marturáis Torres
Concejo 12, 4°A
Tel. 24 25 95 de 2 a 3
32003 Orense

Vendo/Cambio cassette Decathlon y Super Chess (ajedrez) originales por cartucho Konami «Road Fighter» u otras cassettes originales.
Miguel Borreyo
A. Racimir, 11
Olot (Girona)

Cambio MI programa de Karate o Star Avengers por Zaxxon. Envíar Zaxxon contra reembolso a:
José Sánchez Gavilán
Almogavares, 17
Tel. 83 01 80
Llagostera (Gerona)

¡Oferta única! Cambio cartuchos ROM Hyper-Sports I de Konami y Jump Coaster de Sony por Road-Figther de Konami o por Boxing de Konami.
Javier Gil López
Córcega, 60 esc. A 2° 4ª
Tel. (93) 322 26 67
08029 Barcelona

Vendo Master of the Lamps + Phantasma por 1.000 ptas, o cambio Master of the Lamps por Hero, River Raid o Pit Fall II. Envíar cinta a:
Juan Enrique Leoncio
Serrano, 12
Oropesa (Castellón)

Cambio ocho cartuchos Konami primeras marcas (Hiper I, Hiper II, Kung-Fu) por cartucho de ampliación a 64Kb. Para más información llamar a:
Oscar o Pedro José
Tel. (968) 21 38 91

Hit-Bit-Sony-55. Nuevo. Garantía. Programas. Revistas. Ofertas:
José Bua González
Hércules, 105
Tel. (981) 20 10 17
15002 La Coruña

Vendo ordenador Sony HB-75 MSX 80K, como nuevo y económico, por compra de otro MSX con disco, libros de instrucciones en español, cables y regalo programas en cinta, llamar a:
Luis
Tel. (91) 778 52 27
A partir de las 4 de la tarde
Madrid

Vendo ordenador Hit Bit HB-55P. Comprado en Diciembre.
Andoni Rego
Aróstegui, 35
Bermeo (Vizcaya)

Vendo tarjeta Softcard más adaptador. Juego «Jet Set Willy», sólo 48.000 ptas. Está en buenas condiciones.
Pedro J. Gónzalez Fernández
Plaza Neptuno, Bloque 9, 4°B
Tel. 89 75 42
San Fernando (Cádiz)

Urge vender Canon V-20 MSX con poco uso y en perfecto estado. Regalo cables, manuales (muy buenos) y unos 70 programas comerciales de juegos (algunos de aplicación). Lo dejo todo por 50.000 ptas. (preferiblemente de Pontevedra).
Elvis Martínez
Avda. Zamora 99, 2° dcha
Tel. (986) 41 45 25 Vigo 11

Intercambio programas de MSX en discos de 3.5 pulgadas.
Antonio Marín
Garita, 19 ático
Tel.(971) 40 36 59
07015 Palma de Mallorca

Intercambio, programas en código máquina, poseo entre otros: Boulder Dash, Cannon, Norseman, Manes.
Gonzalo Márquez Benítez
Ntra. Sra. de la Merced, s/n
Colegio Aljoxahí
Tel. 25 83 48, tardes desde las 6 p.m.
y demás festivos a cualquier hora
Córdoba

Cambio juegos en cassette con muchachos en Málaga. Llamar a:
Joaquin
Tel. 26 61 18
Málaga

Se vende Sony HB 55P con ampliación de 16K, manuales, cables de conexión y 40 juegos por 25.000 ptas. (a negociar). Llamar o escribir a:
Alvaro Rodríguez
Urzaiz, 81, 7° D
Tel. (986) 41 75 52
Vigo (Pontevedra)

Desearía contactar con usuarios del MSX par poder cambiar programas, opiniones, etc. Llamar a:
Xavier Vilana
Mayor, 19
Tel. 38 31 14 de 9 a 10 de la noche
Organeja (Lérida)

Vendo ordenador SVI-328. Precio a convenir. Se aceptan ofertas.
Javier Sánchez
Tel. 34 06 84
Llamar a partir de las 8,30 en adelante
hasta las 10
Zaragoza

Vendo por cambio de ordenador, Toshiba MSX HX-10 64K (9 meses) y monitor Philips BM 7502 fós. verde (2 meses, casi sin usar, con garantía) por 58.000. Se estudiarán otras ofertas. Opc. juegos, programas y revistas.
Javier
Torras y Pujalt 6
Tel. 248 34 80
Barcelona

Tengo un Toshiba HX-10 y me gustaría cambiar juegos con chicos-as de todas las edades. (Me gustaría un juego de coches). Poseo entre otros, Zaxxon, Manic Miner, Decathlon, etc...
Javier García Valcarce
Marcelo Macías, 13, 8°
Tel. 40 14 94
Ponferrada (León)

Cambio Philips MSX con 48K de Ram y 32 de Rom, 20 cintas, Manuales de uso, todo en perfecto estado, por ZX Spectrum, 48K o vendo por 30.000 ptas.
Juan Ribas Mondejar
Marina, 23
Tel. (972) 84 10 03
Sta. Coloma de Farnes
(Gerona)

Cambio o vendo los siguientes cartuchos: Lode Runner (cientos de pantallas. Juego) Buggy (Coches. Muchas pantallas) Backgamon (Juego de fichas), por programas de utilidades o gestión en cartucho.
Tel. 785 61 77



METODOLOGIA DE LA PROGRAMACION

Autor: A. Martínez, J. Ameller
Editor: Data Becker (Ferre Moret)
Páginas: 256
Precio: 2.200

El objetivo del libro, expresado por los autores en el prólogo del mismo, es el de exponer una metodología para la elaboración de diagramas de flujo, aspecto esencial en la elaboración de un programa.

Con esta intención, los autores desarrollan una serie de temas relacionados con la programación, centrándose sobre todo en la elaboración de diagramas de flujo y en la estructura de un programa (programa principal, subprograma, subrutina). Posteriormente tratan, sin demasiada profundidad, otros aspectos relacionados con la elaboración de programas: organización de ficheros, clasificación y ordenación de datos y utilización de tablas y matrices.

Aunque en toda la obra se dejan entrever aspectos de programación estructurada, es en el capítulo final donde se desarrolla más ampliamente este concepto. En este capítulo se explica incluso un método, el método **Warnier**, para la elaboración de diagramas de flujo y programas que se atengan a las normas de la programación estructurada.

El libro está escrito con una intención fundamentalmente didáctica. Por ejemplo, al final de cada capítulo hay una serie de ejercicios propuestos relacionados con el tema del capítulo. La solución a los mismos aparece en un apéndice de las últimas páginas.

Tanto el lenguaje empleado como el nivel de la obra son asequibles a cualquier lector, aunque no tenga conocimientos previos de programación. Tampoco se requiere ningún conocimiento matemático.

MSX PROGRAMAS Y UTILIDADES

Autor: Rainer Lüers
Editor: Ferre Moret
Páginas: 196
Precio: N.D.



Nos encontramos ante una de las muchas publicaciones que contienen una serie de programas de utilidad/diversión encaminados a que el usuario los pruebe en su ordenador y los incorpore a su bagaje de utilidades propias.

Tras una breve introducción en la que se presenta el libro, y un pequeño apartado dedicado a la compatibilidad entre los ordenadores **Spectravideo 318/328** y los estándard **MSX**, se ofrecen los listados de los programas.

El nivel y la utilidad de los programas ofrecidos es desigual. Frente a algunos de franca utilidad, tales como un editor de gráficos, un editor de sonido, un desensamblador y un gestor de datos, aparecen algunos menos útiles: un calendario, unas tablas deportivas, un medidor de tiempo de reacción, y un descifrador de códigos (que no es más que una versión, eso si, un tanto mejorada, del conocido Master Mind).

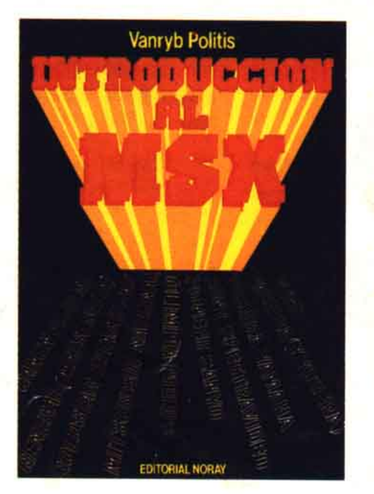
Existen también algunos programas que pueden facilitar la tarea en algunas aplicaciones concretas del BASIC. Entre ellos encontramos un generador de caracteres especiales, de caracteres de ordenador, una ampliación del sistema de gestión de errores del **MSX** y otra de una de una lista de referencia de variables.

Hay que hacer notar, sin embargo, que los comentarios que se incluyen como introducción al comienzo de cada programa podrían haberse enfocado mejor. Aparte de explicar los resultados que proporciona el programa, y la forma de manejarlo, el lector probablemente aprendería más si se insistiera en el programa en sí (conceptos tales como su estructuración, sus bloques, una explicación somera del secuenciamiento, partes, etc.). En nuestra opinión, podrían haberse suprimido algunos de los programas de menos utilidad, y haberse comentado más extensamente los mejores.

INTRODUCCION AL MSX

Autor: Vanryb Politis
Editor: Noray
Páginas: 192
Precio: 1.325 ptas.

Son muchos los libros que pretenden servir de ayuda al principiante en el uso de un ordenador o sistema operativo nuevo para él.



La mayoría de ellos son abandonados por el principiante cuando se familiariza suficientemente con su nuevo equipo, ya que ve que se le quedan cortos.

No es éste el caso de este libro con el que el autor ha pretendido dar una visión suficientemente amplia del sistema **MSX**, partiendo de los conceptos más básicos (que es una variable, o una expresión) y llegando hasta conceptos más complejos como son la gestión del sistema operativo en disco, gráficos, sonido, etc.

Para ello, la obra está dividida en dos partes fundamentales: en la primera,y el autor introduce al lector en el funcionamiento general de los ordenadores del sistema **MSX** y, seguidamente, le enseña a realizar programas en ellos. Es importante resaltar aquí que el método desarrollado es muy ameno evitándose el tradicional de «nombre de la instrucción-explicación de la instrucción-ejemplo», tan frecuentemente usado. Mejor que eso, se explican los elementos fundamentales (constantes, variables, tablas, etc.) y las estructuras fundamentales usadas en programación (condiciones, bifurcaciones, lazos, etc.) desarrollándose su uso con ejemplos sobre programas concretos.

La segunda parte nos ofrece un manual del usuario del BASIC **MSX** tanto en su versión general, como en su versión con sistema operativo en disco, el **MSX-DOS**.

En esta segunda parte el tratamiento es más formal, explicándose detalladamente todos los comandos y haciendo uso de abundantes programas de aplicación. Por ello, esta segunda parte puede constituir por si misma un excelente manual de referencia y consulta para el usuario de un ordenador **MSX**.

Debe destacarse la mención especial que se hace, dedicándose un capítulo a cada uno, de los gráficos y el sonido, herramientas que por el uso particular que se hace de ellas en todo ordenador, merecen un tratamiento especial en toda obra de esta índole.

New Media Systems

NMS

Philips MSX-2

el ordenador multiuso para el hogar y la oficina.

El nuevo Philips MSX-2 es un sistema completo que atraerá a un gran número de personas que usan ordenadores en casa.

Personas tales como el ejecutivo que lleva trabajo a casa, el empleado autónomo, estudiantes y secretarias.

El conjunto entre el avanzado ordenador VG 8235 y nuestro paquete de software, cubren la mayoría de las grandes áreas de aplicaciones productivas. Philips MSX-2 le ofrece un gran sistema a un precio muy atractivo.

El ordenador VG 8235

El primero de la nueva gama de modelos MSX-2, el VG 8235, incorpora una unidad de disco de 3,5" con una capacidad de 360 Kb, 256 Kb RAM, pantalla de 80 columnas y funciones realzadas de color y gráficos.

Interfaces incorporados para impresora, lecto-grabadora y unidad de disco adicional, salida de monitor y TV, conectores de entrada/salida para joysticks, ratón y tableta gráfica y 2 ranuras para cartuchos ROM/RAM.

Paquete de software para la oficina en casa

El software de Philips "Home Office", que acompaña al MSX-2, está separado en 2 paquetes:

MSX Editor: Un paquete de procesador de textos profesional para preparación de alta calidad de todo tipo de documentación, como correspondencia e informes.

MSX Filer: Un programa de base de datos para un rápido y eficiente almacenaje y recuperación de información, tal como nombres, direcciones y números de teléfono.

MSX Editor y MSX Filer pueden usarse en combinación para aplicaciones de correo personalizado o similares.

Además, Philips ofrece un tercer programa con el MSX-2 llamado MSX Designer.

Es un sofisticado paquete de gráficos con Menú-directorio que permite al usuario mezclar color o diseños monocromos con textos, usando el teclado, ratón o tableta de gráficos.

Ascendencia total de compatibilidad MSX

Philips MSX garantiza la total compatibilidad en ascenso, permitiendo que todos los periféricos MSX y software se utilicen con el Philips MSX-2.

Philips MSX-2: El sistema completo para las aplicaciones de la oficina en casa.



Servicio de información al simpatizante y usuario.
Tels. (91) 413 21 61 - 413 22 46

PHILIPS